UNIVERSITÄT PASSAU

INVESTIGATING PRUNED AND STRUCTURAL SPARSITY IN RECURRENT NEURAL NETWORKS

DARJI, HARSHIL JAGADISHBHAI

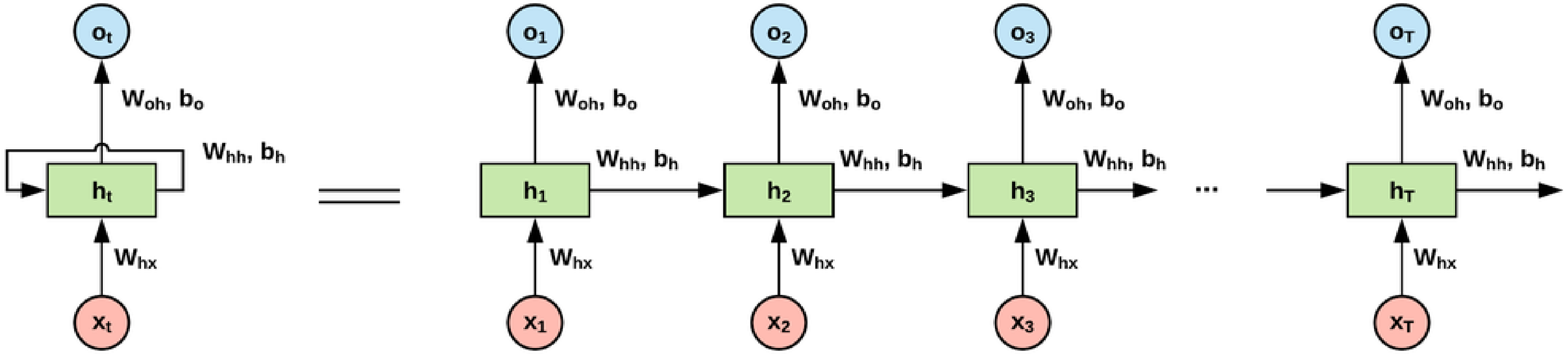
SUPERVISORS:

1. PROF. DR. MICHAEL GRANITZER,
2. JULIAN STIER

AGENDA

1. INTRODUCTION

- Recurrent Neural Networks are standard models that have shown exceptional performance in many NLP tasks that make use of **sequential information**.



Unrolled RNN[1]

- A standard RNN is parameterized with three weight matrices (W_{hx}, W_{hh}, W_{ho}) and two bias vectors (b_h, b_o).
- Deep extensions of such basic RNNs can be constructed by **stacking** multiple recurrent hidden states on top of each other.

[1] Image source: http://www.easy-tensorflow.com/tf-tutorials/recurrent-neural-networks/vanilla-rnn-for-classification

2. MOTIVATION

- Deep Neural Networks are more likely to have increased performance but also **increased fast memory** requirements.
- One way to decrease these memory requirements is to introduce sparsity into a network's connection[1].
- Sparsity in traditional neural networks is being studied widely from the past few years but is not explored much in case of recurrent neural networks.
- Sparse structures have shown a training potential in traditional neural networks[2] which if applied to RNNs, can also make training them **less difficult** while retaining their performance.

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

3. RELATED WORK

- *Exploring Sparsity in Recurrent Neural Networks*[1] *(November 2016) :*
  - Pruned the linear layers that feed into the recurrent layers, the forward and backward recurrent layers and fully connected layer before the CTC layer.
- *Pruned and Structurally Sparse Neural Networks*[2] *(September 2018) :*
  - Tested pruning based sparse topologies by pruning a pre-trained dense network and by using RadiX-Nets.
- *Exploring Randomly Wired Neural Networks for Image Recognition*[3] *(April 2019) :*
  - Explored randomly wired neural networks driven by random graph models from graph theory.
- *Structural Analysis of Sparse Neural Networks*[4] *(September 2019) :*
  - Predicted the performance of convolutional neural networks using its structural properties.

[1] Sharan Narang et al. “Exploring Sparsity in Recurrent Neural Networks”. In: ICLR 2017 Conference. arXiv: 1704.05119v2
[2] Simon Alford et al. Pruned and Structurally Sparse Neural Networks. Tech. rep. MIT, 2018. arXiv: 1810.00299v1
[3] Saining Xie et al. Exploring Randomly Wired Neural Networks for Image Recognition. Tech. rep. Facebook AI Research (FAIR), 2019. arXiv: 1904.01569v2
[4] 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. RESEARCH GOALS

- The primary goal is to investigate, both pruned and structural sparsity in RNNs by answering the following research questions:
  1. How does pruning **impact the RNN's performance** and required training time?
  2. **How much pruning is feasible** before triggering a significant reduction in performance?
  3. How does **structural sparsity in RNNs** compare to standard and pruned RNNs?
  4. Which **graph properties correlate with performance** metrics in structurally 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.

5. EXPERIMENTS

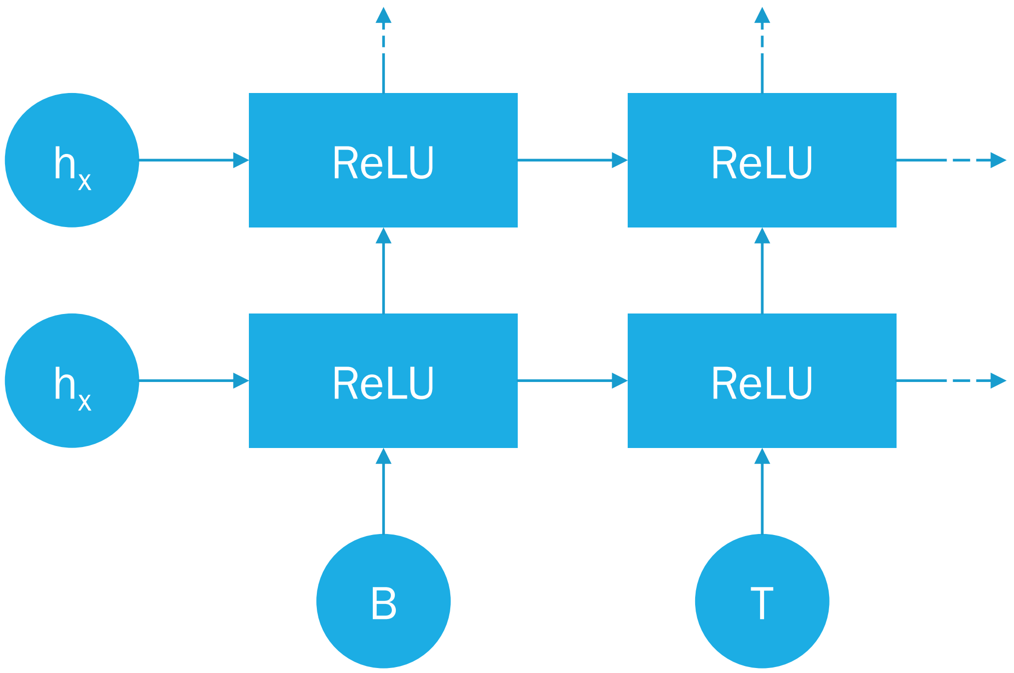
- To investigate the effects of sparsity on the performance of RNNs, three of the experiments, to begin with, are as follow:
  1. Investigate the performance of **pruned RNNs** (*Helps in answering research questions 1 and 2*).
  2. Investigate the performance of **structurally sparse RNNs** (*Helps in answering research questions 3 and 4*).
  3. Investigate the effects of **pruning on structurally sparse RNN** (*Helps in answering research question 5*).
- These experiments are abstractly outlined in the upcoming slides.

5.1. PRUNED RNNS

1. **Train** standard RNNs for a certain epochs,
2. **Retrieve** weight distributions for:
   - Input-to-Hidden matrices,
   - Hidden-to-Hidden matrices,
   - (*Hidden-to-Output matrices?*)
3. **Prune** weights below a certain **threshold** and train again,
4. Retrieve performance and **compare**.

5.1. PRUNED RNNS (cont.)

Suppose we have a Reber sequence: **BTSXSE**



Abstract representation of unrolled RNN for the given sequence

Here,

h_x = Hidden matrices, Initialized with all zeros

ReLU = A single layer with multiple hidden units, (*Not only ReLU, but also TANH/LSTM/GRU*)

Once the pruning is done, hidden matrices are sent **forward** and **to the next layer.**

This process will be repeated for 6 times (*length of the input sequence*).

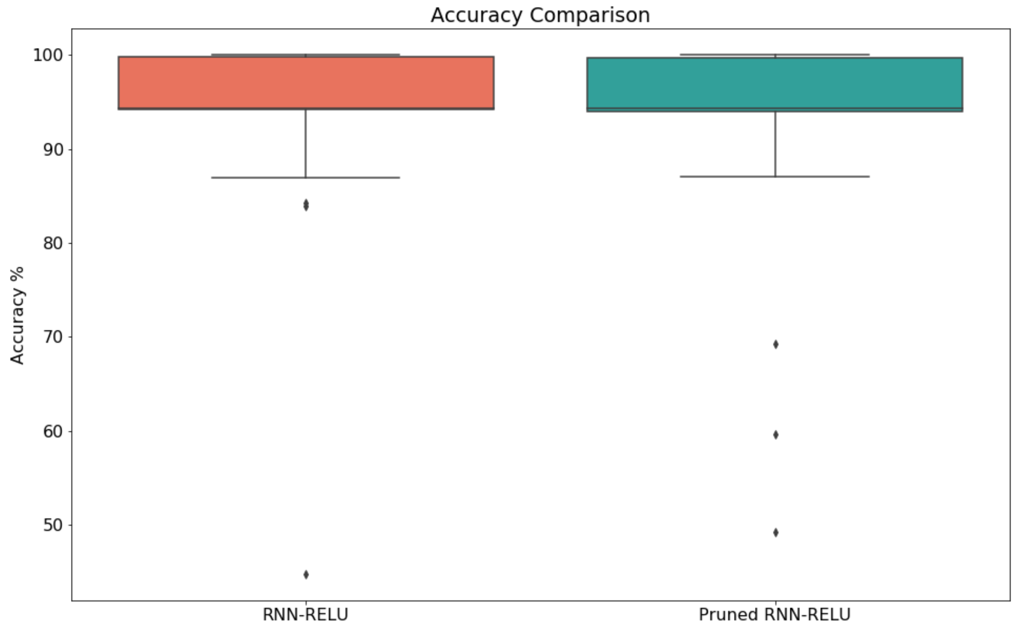
5.1. PRUNED RNNS (cont.)

- Initial results:

I first trained standard RNN with **ReLU** as nonlinearity, pruned hidden matrices, and trained again.

Details about architecture:

Number of hidden layers	3
Hidden units per layer	100 (*For now!*)
Batch size	16
Epochs	12
Learning rate	0.001
Threshold	0.01



Accuracy comparison: RNN-ReLU vs. Pruned RNN-ReLU

Results show that even after pruning hidden matrices, accuracy still stays **approximately the same.**

5.2. STRUCTURALLY SPARSE RNNS

1. Construct sparse feed-forward structures as described in *Structural Analysis of Sparse Neural Networks*[1],
   - Generate random graphs and make them directed,
   - Compute layer indexing of all vertices,
   - Embed these layered vertices between an input and output of an ANN.
2. Introduce **recurrent connections** to make subsequent runs dependent on previous runs,
3. Train sparse RNNs repeatedly for certain epochs,
4. Compare the performance of sparse RNNs with pruned and standard RNNs.

[1] 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

5.3. PRUNING STRUCTURALLY SPARSE RNNS

- This experiment can be considered as a combination of the above two experiments:
  1. Train structurally sparse RNNs for certain epochs,
  2. Retrieve its weight distribution of hidden-to-hidden matrices,
  3. Prune weights below certain levels,
  4. Compare results with structurally sparse RNNs and pruned RNNs.

6. DATASETS

- Depends on the type of RNN being implemented:
  1. **Many-to-One**: Mostly used for classification tasks,
  2. **One-to-Many**: Mainly used for text/sequence generation tasks.
- Initially, while constructing the pipeline, a Reber grammar dataset will be used.
- For this purpose, I have generated a dataset with following characteristics:

Total sequences	25000
Sequence length	≥ 10 characters per sequences
True Reber sequences	12500
Random sequences	12500
Train/Test split	75/25 (shuffled)

7. SCHEDULE

Total
24 weeks

=

Research
4 weeks

+

Implementation
10 weeks

+

Testing
2 weeks

+

Documentation
8 weeks

THANK YOU

QUESTIONS?