Could a Neuroscientist Understand a Recurrent Neural Network?

Joris van Vugt*

January 10, 2017

Abstract

I have applied data analysis methods from neuroscience to an artificial recurrent neural network used for character-level language modelling. Analysis of the activations of single cells revealed a neuron which is responsible for predicting newline characters. It is hard to draw any conclusions about the validity of the methods for neuroscience, since the experiments did not show many insightful results.

1 Introduction

Neuroscientists use various data analysis methods to try to find out how the brain processes information. Jonas and Kording [1] applied some of the popular methods used by neuroscientists to a microprocessor to validate the usefulness of these methods. Although a microprocessor differs from the human brain in many ways, there are some similarities to be found as well. The results from the data analysis should reveal some insights into the true inner workings of a microprocessor in the same way that we think that they provide information about the brain. Since the inner workings of a microprocessor are fully known, the results from the analysis can be compared and validated. Unfortunately, this approach failed to reveal the structure of information processing in a microprocessor. This conclusion would partially invalidate the methodology used in contemporary neuroscience.

The techniques used by Jonas and Kording can also be applied to *artificial neural network* (ANN) models. In particular, the *recurrent neural network* (RNN) is particularly interesting. In contrast to other types of ANN models, an RNN preserves state over time and can therefore be used to model variable-length sequences rather than a fixed input and output format. Arguably, RNNs closer resemble the human brain than a microprocessor and are therefore a better subject for validation of methods from neuroscience.

Firstly, in a microprocessor each component was designed with an exact goal in mind. These components are static and will never adapt over time. In contrast, an RNN is a general purpose system – it can learn to represent and solve many different problems. The brain evolved over time in a somewhat similar fashion: it has learned a representation of the environment that is useful to humans.

Secondly, by using an RNN as a character-level language model, it can be used to generate text [2]. The main advantage of this model is that the output is interpretable. A lot has been learned from patients with a lesion to a particular brain area. The output of a neural network might therefore also be insightful after a "lesion" is applied to it. The microprocessor studied by Jonas and Kording is in that sense limited – a program either works or it crashes.

In the past, Karpathy et al. [3] have found interesting patterns when analyzing the activation of single neurons in a neural network model similar to the one presented here.

*s4279859

2 Methods

2.1 Recurrent Neural Networks

A recurrent neural network (RNN) is an artificial neural network (ANN) model in which the model preserves a (hidden) state vector h over time. This allows the network to make predictions based on information that it has seen previously and create long-term dependencies between input and output. In a simple RNN, the hidden state is updated using a combination of the input x_t and the previous hidden state

$$h_t = \tanh(x_t W_{xh} + h_{t-1} W_{hh} + b_h). \qquad (1)$$

The weights can be optimized by *backpropagation-through-time*. The network is enrolled for a fixed number of time steps and the gradients for all time steps are accumulated. A simple RNN like the one discussed above has 2 major problems however. Repeated multiplication will either cause the gradient to explode or vanish, depending on the values in the weights matrices. The exploding gradient problem can be overcome with a set of simple heuristics (e.g., gradient clipping [4]). The vanishing gradient problem however is more profound.

A RNN called *long short-term memory* (LSTM) has been proposed which solves this problem. The hidden state is modified using explicit input, forget, and output gates. Moreover, another memory state vector c is used. Because of the additive interactions and element-wise multiplications, the problem of vanishing gradient is diminished. An LSTM can be expressed with the following equations:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(x_t W_{xh} + h_{t-1} W_{hh} + b \right) \qquad (2)$$

$$c_t = f \odot c_{t-1} + i \odot g \qquad (3)$$

$$h_t = o \odot \tanh(c_t) \qquad (4)$$

σ is the sigmoid function and is applied element-wise. Element-wise multiplication is denoted by $\odot$. The exact formulation of the activation function doesn't have much impact [5]. Another popular architecture is *gated recurrent unit* (GRU) [6]. However, this does not have an explicit memory state, which is what makes the LSTM interesting for this study.

Multiple recurrent layers can be stacked to create a deeper network. Each subsequent layer will learn more complex representations of the input. In this study, the first layer consists of an LSTM with 256 hidden units. The second layer is an LSTM with 128 units. Finally, the output layer is densely connected and normalized to probabilities using the softmax function.

2.2 Task

In character-level language modelling, a model has to predict the next character given the preceding characters. By feeding the character predicted at the previous time step as input, the model can be used to generate text that is similar to the original dataset [3, 7]. There are two ways to setup this task with an RNN: (1) Given a sequence of characters $x_1,\ldots,x_T$, predict x_{T+1}, and (2) Given a sequence of characters $x_1,\ldots,x_T$, predict each character $x_2,\ldots,x_{T+1}$. These two tasks can be described as sequence-to-one and sequence-to-sequence learning respectively. The latter approach was chosen, since it requires less data and expedites training.

2.2.1 Dataset

The model was trained on *War and Peace* by Leo Tolstoy[1]. All non-ascii characters were converted to their closest ascii equivalent. In total, the datset consists of 3,229,704 characters and 84 unique characters. This is rather small for a language modelling dataset. However, the goal is not to create the best character-level language model, but rather to gain insight into the way an RNN represents such a model.

[1] `http://www.gutenberg.org/ebooks/2600`

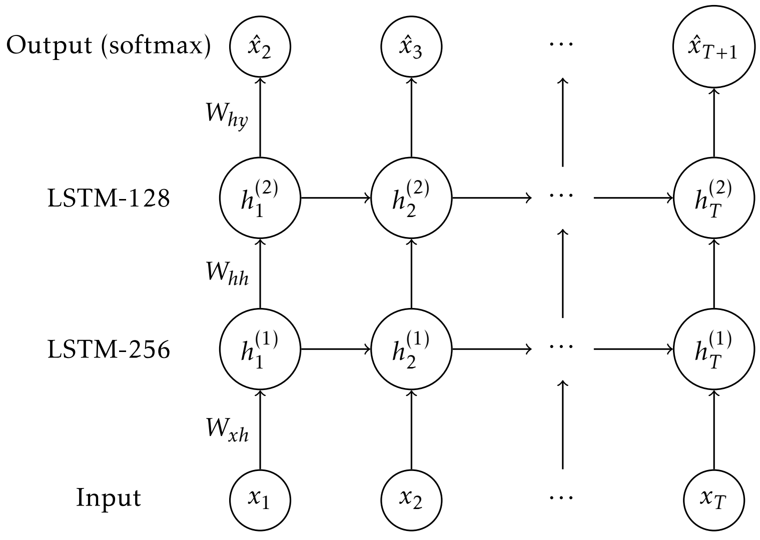
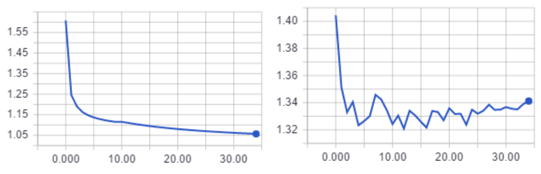


Figure 1: The artificial neural network structure used in this study

2.3 Implementation and Optimization

The network was implemented in *Keras* [8] using the *TensorFlow* [9] back-end. The data was split up into sequences of 100 characters in steps of 7 (i.e., overlapping sequences) and split up into a training set of 90% and 10% validation set. The weights were optimized using RMSProp [10] with a learning rate of 0.002 with a decay of 0.95 after the tenth epoch, which is also used by Karpathy et al. [3]. Furthermore, the second LSTM layer had 20% dropout [11] to control overfitting. After 11 epochs, the model reached a validation set cross-entropy loss of 1.321 (cf. 1.137 test set cross-entropy loss by Karpathy et al.).



(a) Training loss (b) Validation loss

Figure 2: Training process of the network. The best validation loss was reached after 11 epochs (1.321).

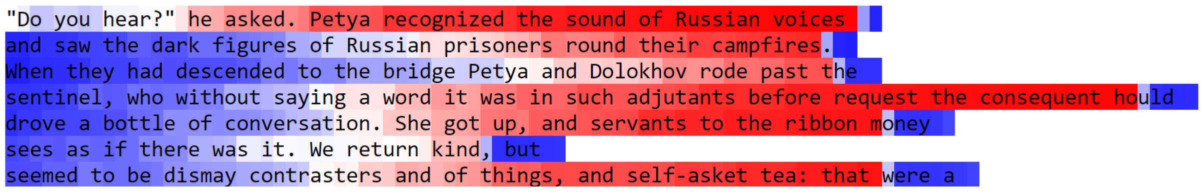
To have better control over the activations in the network for the expiriments, the forward pass of the model was also implemented in *NumPy*. Visualization of the activations was achieved using a custom web-app.

3 Experiments & Results

The experiments are similar to those performed by Jonas and Kording [1] to aid comparability. However, not all experiments make sense in a neural network setting. For instance, each artifical neuron in a layer is connected in the same way as the other neurons in that layer. Because of this symmetry [12], local field potentials don't make a lot of sense. Other experiments, like lesion studies, are reinterpreted in an ANN setting.

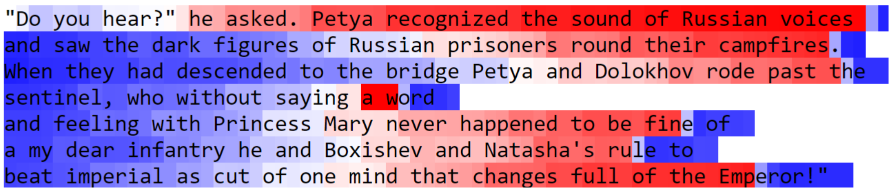
3.1 Visualize Cell States

At each timestep, the cell state vector c_t can be recorded. Figure 3 shows the activation of a single cell when processing and generating text squished by a tanh function (red is positive activation, blue negative activation). The



```
"Do you hear?" he asked. Petya recognized the sound of Russian voices
and saw the dark figures of Russian prisoners round their campfires.
When they had descended to the bridge Petya and Dolokhov rode past the
sentinel, who without saying a word it was in such adjutants before request the consequent hould
drove a bottle of conversation. She got up, and servants to the ribbon money
sees as if there was it. We return kind, but
seemed to be dismay contrasters and of things, and self-asket tea: that were a
```

(a) All text after the third line is generated by the network



```
"Do you hear?" he asked. Petya recognized the sound of Russian voices
and saw the dark figures of Russian prisoners round their campfires.
When they had descended to the bridge Petya and Dolokhov rode past the
sentinel, who without saying a word
and feeling with Princess Mary never happened to be fine of
a my dear infantry he and Boxishev and Natasha's rule to
beat imperial as cut of one mind that changes full of the Emperor!"
```

(b) The neuron is artificially excited after *"a"* on the fourth line

Figure 3: A neuron which gets more excited the longer the line is. This is useful in predicting a newline character. In panel (b), the neuron is artificially excited on the fourth line. As to be expected, a newline immediately follows *"word"*, whereas the line continued in panel (a).

cell shown here is responsible for keeping track of the line length. It gradually gets more excited for each character and could be useful in predicting a newline character. Unfortunately, I was able to only find one interpretable cell, despite trying many different network configurations. Most other cells were indistinguishable from random noise.

3.2 Lesion studies

Lesion studies have proved to be very insightful in neuroscience. Analogous, a cell in an RNN can be artificially inhibited or excited. Figure 3b shows that when this particual cell is manually excited, a newline will likely follow very quickly. This confirms the earlier belief that this cell is at least partially responsible for predicting a newline character. This result was consistent across trials.

3.3 Dimensionality Reduction

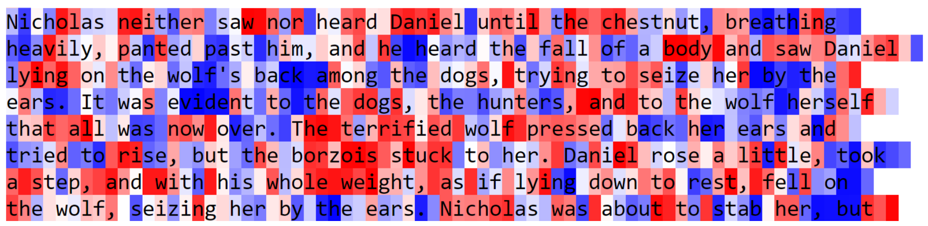
Lastly, I have applied dimensionality reduction to the cell activations. Jonas and Kording [1] used *non-negative matrix factorization* (NMF). This technique can not be applied in this case, since the cell activations can be negative. Instead, *principal component analysis* (PCA) was performed on cell activations from 10,000 timesteps. Unfortunately, no component showed interpretable results regardless of the number of components. Figure 4 shows one of 50 components from PCA.

4 Conclusion

Visualization of the cell states has given some insight into the workings of an RNN and this result could be validated using a lesion study. Unfortunately, the number of interpretable cells was not as numerous as hoped. Dimensionality reduction failed to give any meaningful results. It is thus hard to draw a conclusion about the application of neuroscientific methods to RNNs. In the next section I will give some pointers to hopefully improve these results.

5 Future Work

Firstly, a better network could be used. Because of constraints on compute power and time, the



```
Nicholas neither saw nor heard Daniel until the chestnut, breathing
heavily, panted past him, and he heard the fall of a body and saw Daniel
lying on the wolf's back among the dogs, trying to seize her by the
ears. It was evident to the dogs, the hunters, and to the wolf herself
that all was now over. The terrified wolf pressed back her ears and
tried to rise, but the borzois stuck to her. Daniel rose a little, took
a step, and with his whole weight, as if lying down to rest, fell on
the wolf, seizing her by the ears. Nicholas was about to stab her, but
```

Figure 4: The value of one of 50 components from principal component analysis at each timestep. All components showed behaviour similar to the one showed here and were difficult to interpret.

size of the network used in this study is relatively small. Moreover, the optimization of the network could be done more carefully to achieve better results. For example, the results achieved by Karpathy et al. [3] with a similar network were significantly better.

Secondly, a dataset with more structure could provide more insights. Structured text, such as source code, has more patterns than natural language which can be picked up by the RNN.

Thirdly, a larger dataset could be used to achieve a better model. Structured text can already be a lot longer than a book like War and Peace. A larger corpus of natural language will also be useful, although performance may suffer if the text is too heterogeneous.

Lastly, more experiments could be cast into an RNN setting. For example, local field potential could be interpreted in the context of layers in a neural network.

6 References

[1] Eric Jonas and Konrad Kording. Could a neuroscientist understand a microprocessor? *bioRxiv*, page 055624, 2016.

[2] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[3] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[4] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013.

[5] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

[6] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[7] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[8] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[9] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin

Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[10] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[11] Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. *arXiv preprint arXiv:1512.05287*, 2015.

[12] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.