Visualizing and Understanding Recurrent Networks

Andrej Karpathy*, Justin Johnson*, Fei Fei Li

International Conference on Learning Representations, 2016

Problems Addressed

1. The paper aims to further the understanding of how LSTMs work, by providing an analysis of their representations, predictions and error types.
2. The analysis done by earlier papers shed light only done on the performance critical pieces of architecture, and that too, on the level of final test set perplexity alone.
3. The paper also aims to shed light on the fact that LSTMs can store and retrieve information over long time ranges on real world test data with the common use of simple stochastic gradient descent and truncated backpropagation through time, as this property has only been studied in toy settings till now.

Proposed Solution

1. Character level language models are used to provide an interpretable platform for identifying long term dependencies in an LSTM network.
   - In this, the network is fed with character-level input, and it is trained to predict the next character in the sequence with a softmax classifier.
   - Assuming a vocabulary of K characters, the inputs are defined by one hot vectors of size K.
   - These, when fed into the network, lead to D dimensional hidden state vectors h_t.
   - To get the output from the state of the RNN model, the final layer is projected onto a sequence of K dimensional vectors $\mathrm{y_t}$.

   $$y_t = W_y h_t^L,\ where\ W_y\ is\ a\ K \times D\ matrix$$

   These vectors are interpreted as containing the unnormalized log probabilities of the next character and hence, the objective is to minimize the cross-entropy loss over all targets.
2. All parameters are initialized uniformly in the range of [-0.08, 0.08]. Mini-batch SGD with a batch size of 100 and RMSProp per parameter adaptive update with base learning rate of 2×10^{-3} and decay 0.95 are used. The settings remain same across all models.
3. The network is unrolled for 100 time steps. Every model is trained for 50 epochs, with the learning rate being decayed after 10 epochs, by multiplying with a factor of 0.95 after each additional epoch.
4. Datasets used are the novel *War and Peace* with 3,258,246 of mostly English characters and minimal markup (with data split in 80:10:10 ratio of Training:Validation:Testing), and the source code of *Linux Kernel*, with header and source files shuffled randomly and concatenated leading to 6,206,996 character long file (with split ratio being 90:5:5). **These datasets were chosen above others like Penn Treebank because the aim is to study the working in controlled settings.**
5. The total number of characters in vocabulary was 87 for *War and Peace* and 101 for *Linux Kernel*.

Experiments

1. Each model {RNN/LSTM/GRU} is trained on both the datasets separately with multiple settings {Layers (1/2/3) and number of Parameters (4 settings)}.
   - For 1 Layer LSTM : Hidden Vectors of sizes 64/128/256/512, with character vocabulary sizes being approximately 50K, 130K, 400K and 1.3M respectively.
   - The sizes of hidden layers in other models were chosen so as to keep the total number of parameters as close as possible in all 4 settings (values not revealed in paper).

Results

	LSTM			**RNN**			**GRU**		
Layers	1	2	3	1	2	3	1	2	3
Size	War and Peace Dataset								
64	1.449	1.442	1.540	1.446	1.401	1.396	1.398	**1.373**	1.472
128	1.277	1.227	1.279	1.417	1.286	1.277	1.230	**1.226**	1.253
256	1.189	**1.137**	1.141	1.342	1.256	1.239	1.198	1.164	1.138
512	1.161	1.092	1.082	-	-	-	1.170	1.201	**1.077**
	Linux Kernel Dataset								
64	1.355	**1.331**	1.366	1.407	1.371	1.383	1.335	1.298	1.357
128	1.149	1.128	1.177	1.241	**1.120**	1.220	1.154	1.125	1.150
256	1.026	**0.972**	0.998	1.171	1.116	1.116	1.039	0.991	1.026
512	0.952	0.840	0.846	-	-	-	0.943	0.861	**0.829**

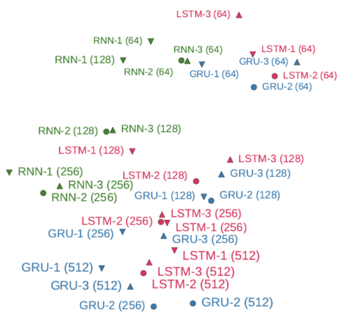

Figure 1: **Left:** The **test set cross-entropy loss** for all models and datasets (low is good). Models in each row have nearly equal number of parameters. The test set has 300,000 characters. The standard deviation, estimated with 100 bootstrap samples, is less than 4×10^{-3} in all cases. **Right:** A t-SNE embedding based on the probabilities assigned to test set characters by each model on War and Peace. The color, size, and marker correspond to model type, model size, and number of layers.

1. **Depth and Prediction Similarities**
   - Depth of at least 2 was found to be beneficial; mixed results between depths of 2 and 3 (see Figure 1).
   - Also computed the fraction of times the models agreed on the most likely character and used it to render a t-SNE embedding (it was found to be more robust than KL divergence) which reinforces the fact that LSTMs and GRUs made similar predictions while RNNs form their own cluster.
2. **Interpretable, Long Range LSTM cells**
   - Found a cell which acts as a line length counter, as it starts with a high value and slowly decays till the next newline character.
   - Similarly found other cells which turn on inside quotes, if statements, parentheses or with increasing strength as indentation of a block of code increases. Therefore, even though the backpropagation is truncated to not go beyond 100 time steps, such switches allow the network to effectively predict the closing quote or parenthesis even after more than 100 timesteps.
   - Therefore, LSTM cells are capable of using such operations despite practical optimization challenges (SGD dynamics and truncated gradient signals).

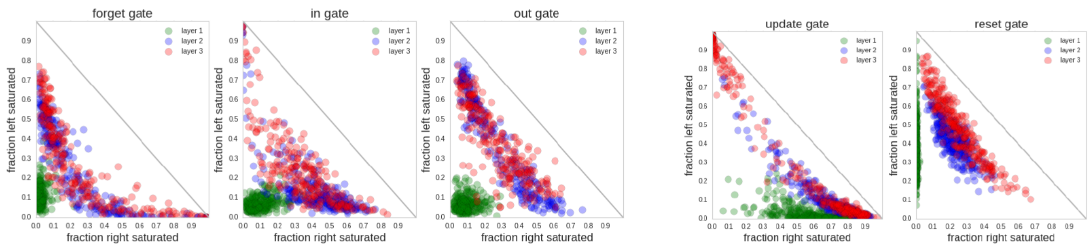

Figure 3: **Left three:** Saturation plots for an LSTM. Each circle is a gate in the LSTM and its position is determined by the fraction of time it is left or right-saturated. These fractions must add to at most one (indicated by the diagonal line). **Right two:** Saturation plot for a 3-layer GRU model.

3. Gate Activation Statistics

- Defined a gate to be left or right saturated if $y \leq 0.1$ *or* $y \geq 0.9$ respectively, where y is the activation value. The fraction of time that a neuron spends left or right saturated was then computed.
- Right saturated **forget gates** correspond to the cells that remember their values for long durations. Also notice that there are no cells that are purely feed-forward in fashion, as no forget gate is consistently left saturated.
- **Output gate** statistics reveal that no cell is consistently revealed or blocked to the hidden state.
- The activations in the first layer are very diffuse, unlike the other layers where the gates are frequently left or right saturated. (No explanation available as of now, but this is seen across all models). This points towards a purely feedforward mode of operation on this layer, where previous hidden state values are rarely used.

4. Further testing Long-Range Interactions

- Here, LSTM is compared with baseline models (n-NN: a fully connected NN with tanh activations, with an input vector of size nK that concatenates one hot vectors of n consecutive characters; n-gram: An unpruned n+1 gram language model, trained using KenLM software package) that can only utilize information from a fixed number of previous steps.
- Both baseline models perform similarly for small n, but for larger n the NN starts overfitting and n-gram performs better. Also, the best recurrent network outperforms the 20 gram model, providing weak evidence that the network is effectively utilizing information from beyond 20 characters (weak as the assumptions encoded in Kneser-Ney smoothing model are intended for word-level modelling).
- Error analysis (With error defined as probability of next character < 0.5): It is found that majority of errors are shared by all three models, but each model also has its own unique errors. LSTMs display great advantage over special characters.

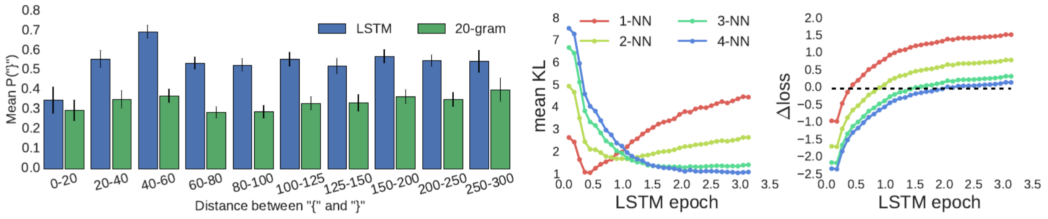

Figure 5: **Left**: Mean probabilities that the LSTM and 20-gram model assign to the "}" character, bucketed by the distance to the matching "{". **Right**: Comparison of the similarity between 3-layer LSTM and the n-NN baselines over the first 3 epochs of training, as measured by the symmetric KL-divergence (middle) and the test set loss (right). Low KL indicates similar predictions, and positive Δloss indicates that the LSTM outperforms the baseline.

- Considering closing brace, LSTM only slightly outperforms 20-gram when distance between braces is ≤ 20 characters, but after this point the performance of 20-gram stays constant (reflecting baseline probability of predicting closing brace without seeing the opening brace) while LSTM gains significant boosts up to 60 characters, and then its performance declines as it becomes harder to remember dependencies (Figure 5).

5. **Training Dynamics**

   - During training, compared the LSTM with trained n-NN models using KL divergence between the predictive distributions on test set.
   - It is found that during the first few iterations, LSTM behaves like 1-NN, but then soon diverges, behaving more like 2, 3 and 4-NN models in turn. This suggests that LSTM "grows" its competence over increasingly longer dependencies during training. **This might be the reason why Sutskever et al. (2014) observed improvements on reversing the input sequence - it introduces short term dependencies that the LSTM can model first, and the longer dependencies are learned over time**.

6. **Error Analysis**

   Ordering of oracles can lead to different results. The authors tried to apply oracles in order of increasing difficulty of removing each error category.
   The authors then subject their best LSTM model and the best LSTM model with 50K parameters to error analysis, to check how oracles affect them individually and how scaling affects the errors.

   - **n-gram oracle:** Modelled optimistic n-gram oracles with $n \in \{1..9\}$ which remove errors that can be eliminated by better modelling short term dependencies.
   - **Dynamic n-long memory oracle:** Consider the string "Jon yelled at Mary but Mary couldn't hear him."
     - One consistent failure mode that the authors noticed in the predictions is that if the LSTM fails to predict the characters of the first occurrence of "Mary" then it will almost always also fail to predict the same characters of the second occurrence, with a nearly identical pattern of errors.
     - In principle, the first mention should make the second more likely, as the LSTM could store a summary of previously seen characters and fall back on this information when it is uncertain. **However, this does not appear to take place in practice**.
     - This *dynamic aspect* is a common feature of sequence data, where certain subsequences that might not appear frequently in training data should still be more likely if they were present in immediate history.

     This oracle quantifies the severity of this limitation by removing errors in all words (starting with second character) that can be found as a substring of last n characters, with $\mathrm{n} \in \{100, 500, 1000, 5000\}$.
   - **Rare words oracle:** Removes errors for rare words that occur only upto n times in the training set, with $\mathrm{n} \in 0, ..., 5$.
   - **Word model oracle:** It was noticed that a large portion of error occur in the first letter of each word, as it is easier to complete a known word that select the next word. Therefore, this oracle removes all errors after space, quote or newline.
   - The remaining errors cannot be easily pinpointed on any one interpretable aspect. Hence, the authors introduce a **punctuation oracle** (removes punctuation errors) and a **boosting oracle** (boosts probability of correct letter by fixed amount).

   The best LSTM made 140K errors out of 330K test set characters - out of these, n-gram eliminates 18% errors, dynamic memory oracle removes 6% errors, rare words oracle removes 9% of errors. 37% of remaining errors are due to difficulty with word-level predictions.
   The smaller model made 184K errors. Out of the extra 44K errors, 81% were n-gram errors, 11% came from boost category, and the remaining were evenly distributed amongst other categories. **This indicates that better architectures may be required to reduce other errors instead of simply scaling up the model**.

Bibliography

1. Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

2. Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on, 5(2):157–166, 1994.

3. Chen, Stanley F and Goodman, Joshua. An empirical study of smoothing techniques for language modeling. Computer Speech & Language, 13(4):359–393, 1999.

4. Cho, Kyunghyun, van Merri¨enboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

5. Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

6. Dai, Andrew M and Le, Quoc V. Semi-supervised sequence learning. arXiv preprint arXiv:1511.01432, 2015.

7. Dauphin, Yann N, de Vries, Harm, Chung, Junyoung, and Bengio, Yoshua. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. arXiv preprint arXiv:1502.04390, 2015.

8. Donahue, Jeff, Hendricks, Lisa Anne, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. CVPR, 2015.

9. Graves, Alex. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

10. Graves, Alex, Mohamed, A-R, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pp. 6645–6649. IEEE, 2013.

11. Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural turing machines. arXiv preprint arXiv:1410.5401, 2014.

12. Greff, Klaus, Srivastava, Rupesh Kumar, Koutn´ık, Jan, Steunebrink, Bas R., and Schmidhuber, J¨urgen. LSTM: A search space odyssey. CoRR, abs/1503.04069, 2015. URL http://arxiv. org/abs/1503.04069.

13. Heafield, Kenneth, Pouzyrevsky, Ivan, Clark, Jonathan H., and Koehn, Philipp. Scalable modified Kneser-Ney language model estimation. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pp. 690–696, Sofia, Bulgaria, August 2013. URL http://kheafield.com/professional/edinburgh/estimate_paper.pdf.

14. Hermans, Michiel and Schrauwen, Benjamin. Training and analysing deep recurrent neural networks. In Advances in Neural Information Processing Systems, pp. 190–198, 2013.

15. Hochreiter, Sepp and Schmidhuber, J¨urgen. Long short-term memory. Neural computation, 9(8): 1735–1780, 1997.

16. Hoiem, Derek, Chodpathumwan, Yodsawalai, and Dai, Qieyun. Diagnosing error in object detectors. In Computer Vision–ECCV 2012, pp. 340–353. Springer, 2012.

17. Huang, Xuedong, Acero, Alex, Hon, Hsiao-Wuen, and Foreword By-Reddy, Raj. Spoken language processing: A guide to theory, algorithm, and system development. Prentice Hall PTR, 2001.

18. Hutter, Marcus. The human knowledge compression contest. 2012.

19. Jelinek, Frederick, Merialdo, Bernard, Roukos, Salim, and Strauss, Martin. A dynamic language model for speech recognition. In HLT, volume 91, pp. 293–295, 1991.

20. Joulin, Armand and Mikolov, Tomas. Inferring algorithmic patterns with stack-augmented recurrent nets. CoRR, abs/1503.01007, 2015. URL http://arxiv.org/abs/1503.01007.

21. Jozefowicz, Rafal, Zaremba, Wojciech, and Sutskever, Ilya. An empirical exploration of recurrent network architectures. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pp. 2342–2350, 2015.

22. Karpathy, Andrej and Fei-Fei, Li. Deep visual-semantic alignments for generating image descriptions. CVPR, 2015.

23. Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of english: The penn treebank. Computational linguistics, 19(2):313–330, 1993.

24. Mikolov, Tom´aˇs. Statistical language models based on neural networks. Presentation at Google, Mountain View, 2nd April, 2012.

25. Mikolov, Tomas, Karafi´at, Martin, Burget, Lukas, Cernock‘y, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010, pp. 1045–1048, 2010.

26. Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. arXiv preprint arXiv:1211.5063, 2012.

27. Pascanu, Razvan, G¨ulc¸ehre, C¸ aglar, Cho, Kyunghyun, and Bengio, Yoshua. How to construct deep recurrent neural networks. CoRR, abs/1312.6026, 2013. URL http://arxiv.org/abs/1312.6026.

28. Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

29. Schmidhuber, J. Deep learning in neural networks: An overview. Neural Networks, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003. Published online 2014; based on TR arXiv:1404.7828 [cs.NE].

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

31. Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc VV. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pp. 3104–3112, 2014.

32. Van der Maaten, Laurens and Hinton, Geoffrey. Visualizing data using t-sne. Journal of Machine Learning Research, 9(2579-2605):85, 2008.

33. Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. CVPR, 2015.

34. Werbos, Paul J. Generalization of backpropagation with application to a recurrent gas market model. Neural Networks, 1(4):339–356, 1988.

35. Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks. CoRR, abs/1410.3916, 2014. URL http://arxiv.org/abs/1410.3916.