Grid LSTM

Eric Chu
MAS.S63: New Destinations in Artificial Intelligence

1 Background

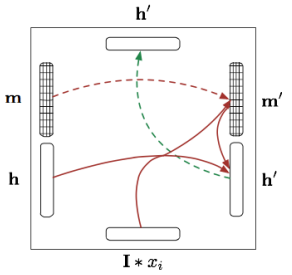
Inspired by my presentation on the Neural Random-Access Machine (NRAM) and computational models of cortical function, I wanted to tackle a more complex neural network architecture. As impressive as deep neural networks have been on a number of tasks in computer vision, speech recognition, and natural language processing, they appear to be as of yet missing components that can lead to higher order cognitive functions such as planning and conceptual reasoning. Moreover, it seems natural to assume that more sophisticated reasoning and learning can be achieved as we move past our relatively simple conception of neural networks as a series of stacked layers.

While time constraints led me to ultimately settle on an architecture that is perhaps slightly less interesting and less cognitively-inspired than I would have liked, it was still nice tackling one that appears to do well on both more synthetic tasks such as the algorithmic replication done in the Neural Turing Machine and NRAM papers, as well as real-life tasks such as language modeling and machine translation [3].

2 Long short-term memory

Motivation. The simpler recurrent neural network (RNN) architecture suffers from two main problems during training: the vanishing gradient problem and the exploding gradient problem. In the vanishing gradient problem, for instance, the gradient of a non-linearity such as the sigmoid function is equal to zero outside of a small region around $x = 0$. Consequently, the errors are not backpropagated properly, and the network fails to learn [5].

Formulation. The long short-term memory (LSTM) network addresses these issues by creating a pipeline in the network through which information can be passed from time step to time step through linear transformations. In addition to the hidden vector h present in a RNN, a LSTM also has a memory cell vector m. What gets stored in these two vectors across time steps is modulated by four gates within each LSTM block. The *forget gate* determines which parts of the memory vector to delete, the *input gate* determines which parts of the memory vector to update, the *content gate* determines what values the memory vector should be updated with, and the *output gate* determines what gets read from the new memory into the new hidden vector. Each gate takes as input a non-linearity and some function of the input at that timestep x_i and the previous hidden vector h. Each gate is defined in the following, and one LSTM block is shown in Figure 1. Here, $H = [Ix_i; h]$, where Ix_i can be a projection layer(s) creating an embedding for the raw input x_i.



Standard LSTM block

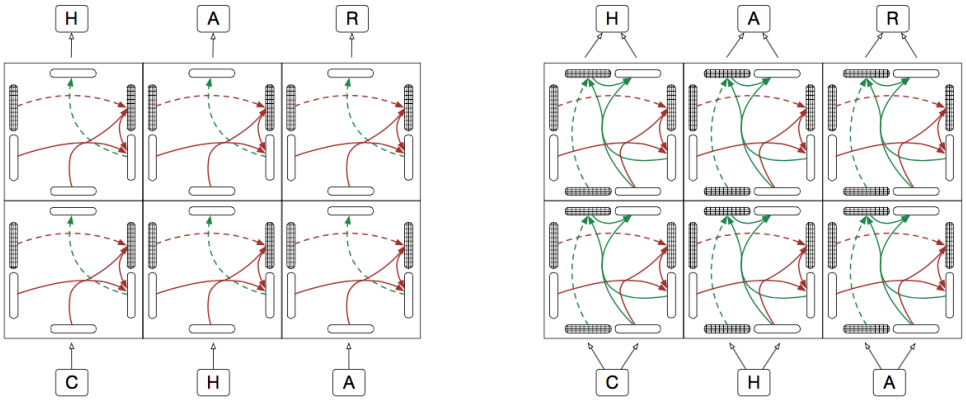
Figure 1

$$
\begin{aligned}
g^u &= \sigma(W^u H) \\
g^f &= \sigma(W^f H) \\
g^o &= \sigma(W^o H) \\
g^c &= tanh(W^c H) \\
m' &= g^f \odot m + g^u \odot g^c \\
h' &= tanh(g^o \odot m')
\end{aligned}
$$

3 Grid LSTM

Motivation. Quite often, LSTM's are stacked such that the hidden layer at a given timestep h_t, is the input for the next layer of the LSTM. This is shown in in the left hand side of Figure 2. The 2D Grid LSTM simply extends this architecture by proposing that the LSTM memory cells and gates should extend to the vertical depth dimension as well as the horizontal temporal dimension [2].

Generalizations. Grid LSTM can be extended to an N-D grid. In the 1D case, this is very similar to the recent Highway network architecture and uses LSTM cells instead of regular non-linearities in a feed forward neural network. In the 3D case, the Grid LSTM is a generalization of multidimensional LSTMs, which are commonly are used on 2D data such as images. Multidimensional LSTMs have a third depth dimension for stacking, and the 3D Grid LSTM simply extends this by having LSTM cells along this depth dimension. The N-D Grid LSTM has N inputs and N outputs at each LSTM block, as can be seen in Figure 3 [1].



Stacked LSTM 2d Grid LSTM

Figure 2

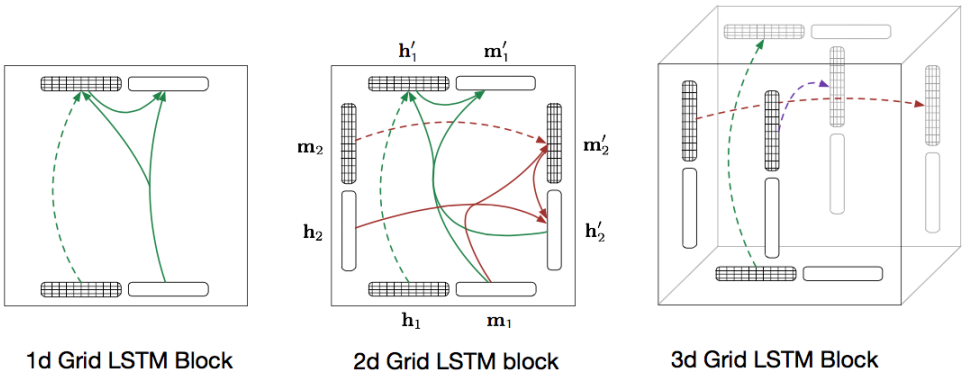


Figure 3

4 Implementation

The Lua-based Torch7 library was used to implement the model. I followed the char-rnn approach to implementing recurrent networks, which is to have T clones of the network for a sequence of length T. This is done so that each clone has its own "output" and "gradInput" (gradient with respect to the input) state variables so that we can backpropagate through time.

An interesting detail noted in the paper that becomes immediately apparent is that dimensions can be prioritized. As the hidden vectors of each dimension are updated sequentially, an updated hidden vector can be used to compute the update of a second hidden vector. Specifically, if our goal is to prioritize the first dimension of the network, the LSTM block will set H equal to $[h1; h'_2; ...; h'_N]$.

5 Experiment: Character Prediction

To test the effectiveness of the Grid LSTM, I compared the 2D Grid LSTM to a regular LSTM on a character prediction task. Figure 4 shows that the 3 layer Grid LSTM has a lower validation loss than the 3 layer stacked LSTM.

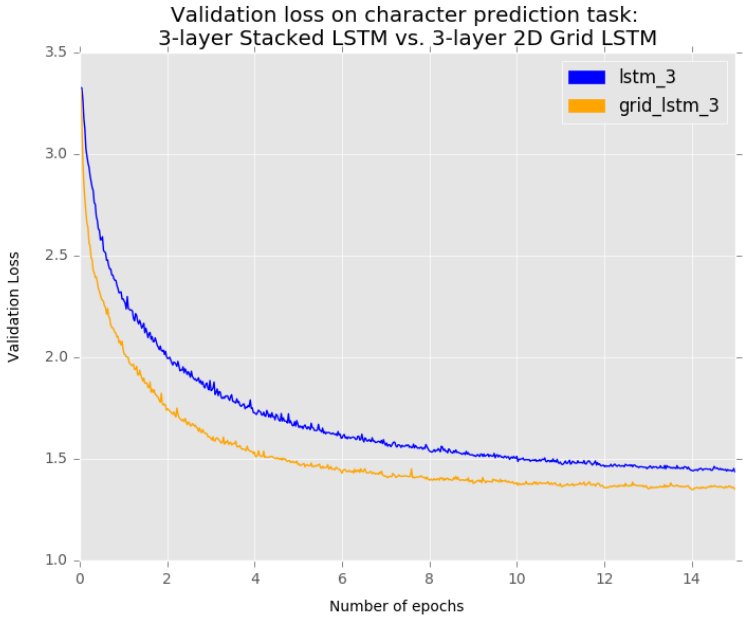


Figure 4

6 Discussion and Next Steps

While not as explicit as an attention network such as DRAW, [4] describes a few connections between LSTMs and attention. For instance, LSTM can be thought of as an implicit attention system that can attend to past information by retrieving it from the memory cell and placing it in the hidden vector. In another sense, the computational view of existing attention models is that it allows for a total of $O(T^2)$ computation. However, one achieves the same result by placing the input and the output along two dimensions in a Grid LSTM. To be honest, I'm just looking for connections to attention because it's an area I would like to keep exploring in the future.

I still hope to apply this to my research in speech recognition and language modeling, but I also hope that in the spirit of this class, I keep pursuing ideas that, to quote Joscha, "won't get you tenure".

References

[1] Graves, Alex, and Jürgen Schmidhuber. "Offline handwriting recognition with multidimensional recurrent neural networks." Advances in neural information processing systems. 2009.

[2] Kalchbrenner, Nal, Ivo Danihelka, and Alex Graves. "Grid long short-term memory." arXiv preprint arXiv:1507.01526 (2015).

[3] Kurach, Karol, Marcin Andrychowicz, and Ilya Sutskever. "Neural Random-Access Machines." arXiv preprint arXiv:1511.06392 (2015).

[4] Le, Quoc V. "A Tutorial on Deep Learning Part 2: Autoencoders, Convolutional Neural Networks and Recurrent Neural Networks." (2015).

[5] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks." arXiv preprint arXiv:1211.5063 (2012).