Danijar Hafner

Layer Norm and GRU for State of the Art Language Modeling

Aug 29, 2017

I conducted some experiments to see what types of recurrent neural networks work well on the Text8 dataset. Specifically, I compared the memory cells LSTM, GRU, and MGU, whether to use layer normalization, and three methods for initializing weights.

Suprisingly, this gives you state-of-the-art performance without using any of the recent regularization methods. If you want to try this out yourself, here is my TensorFlow implementation of the GRU cell that supports layer normalization and the different initializers.

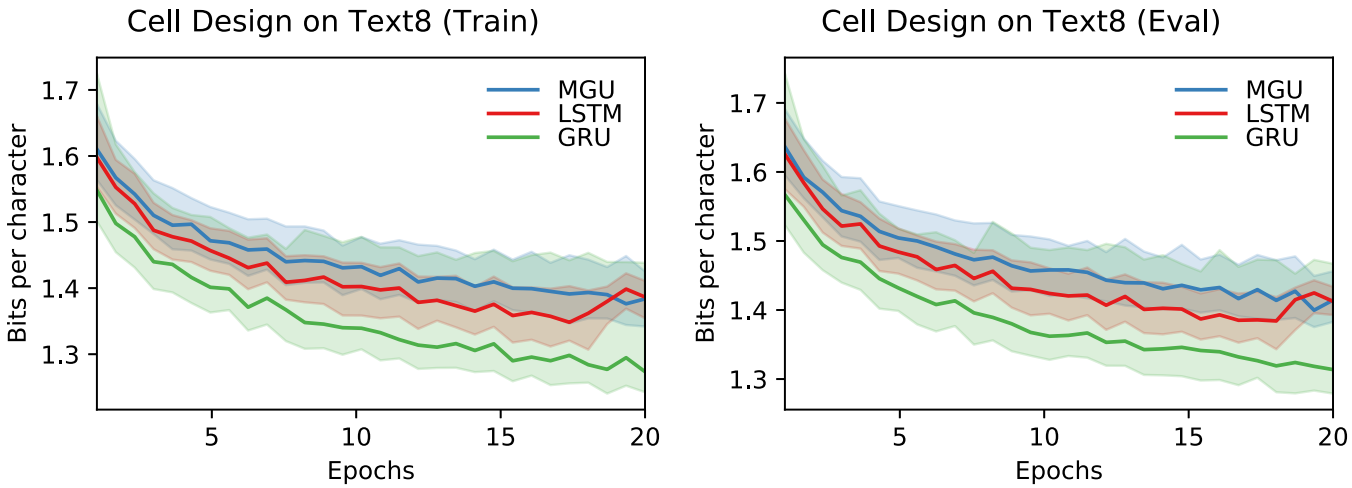
Text8 is a language modeling benchmark, where the neural network reads part of a Wikipedia article and needs to predict the following character of the text. Performance is measured in bits per character (BPC), which describes how well our model can compress the text. The fewer bits, the better.

All experiments use a single recurrent layer of 2000 units, a batch size of 100, sequences of length 200 bytes, and the Adam optimizer with a fixed learning rate of 10^{-3}.

Memory Cell Design

Plain recurrent neural networks compute a completely new hidden state at every time step. This makes it hard for them to remember details over many time steps. The most common solution for this is the LSTM cell, that uses local context values that are preserved over time steps.

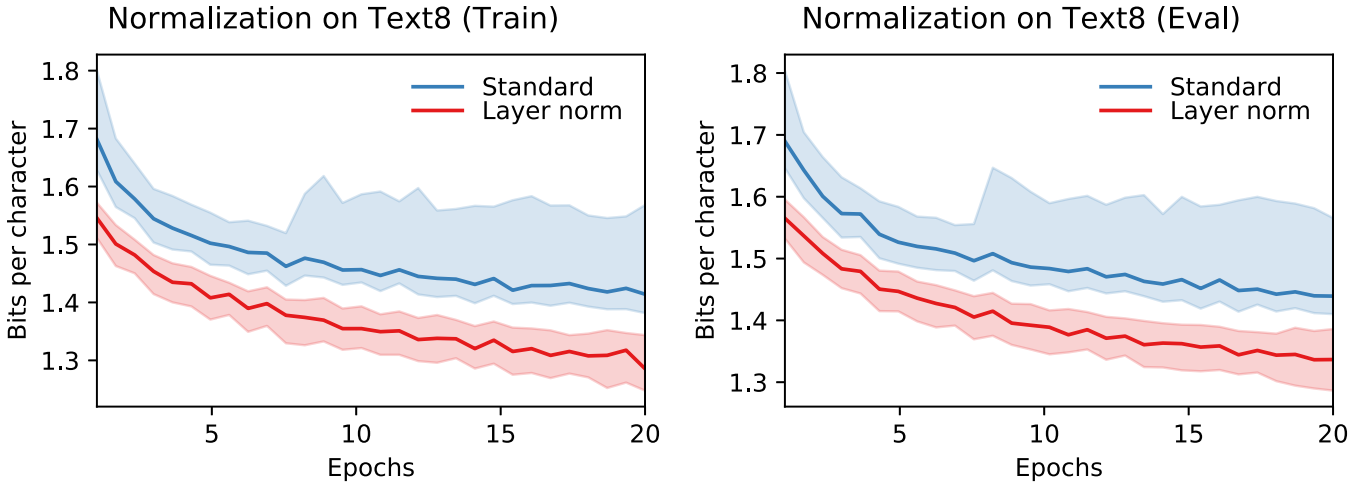
In the last years, researcher proposed several alternatives to reduce the complexity and number of parameters of LSTM. Here, I'm comparing Long Short-Term Memory (LSTM, Hochreiter97) to the Gated Recurrent Unit (GRU, Chung15) and the Minimal Gated Unit (MGU, Zhou16).



Interestingly, GRU performs better than LSTM here, although it uses fewer parameters. Usually, more parameters is a big advantage in compression tasks such as language modeling. MGU uses the fewest parameters and also performs worst on this task.

Layer Normalization

Normalization inside neural networks is known to improve performance in many cases. Especially recurrent networks suffer from vanishing or exploding gradients when their weight matrix changes the magnitude of hidden activations too much between time steps. Layer norm (Ba16) centers and scales the activations at every time step, so that they stay in a similar range.

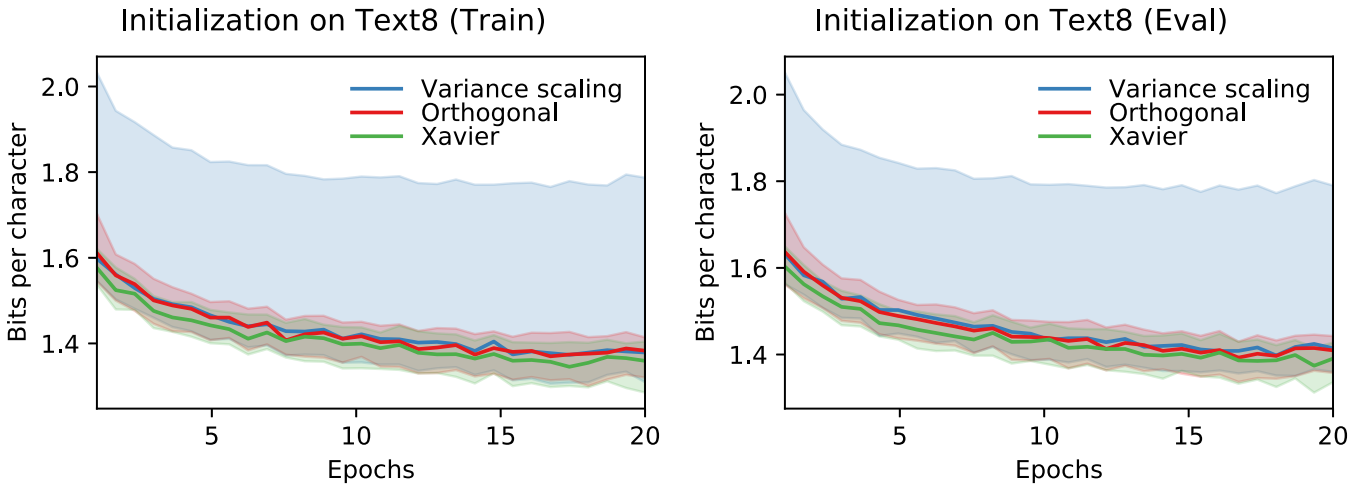


The results shown here are averaged along the different memory cell designs and weight initializations. As seen in the diagram, layer norm both speeds up training and results in significantly better final performance. I am surprised by such clear results, but at least from this task it looks like layer norm should be a default for recurrent networks.

Weight Initialization

Sometimes, the way we initialize weights is critical in training neural networks. There are several methods for this, that basically sample weights from different distributions and scale them based on layer sizes.

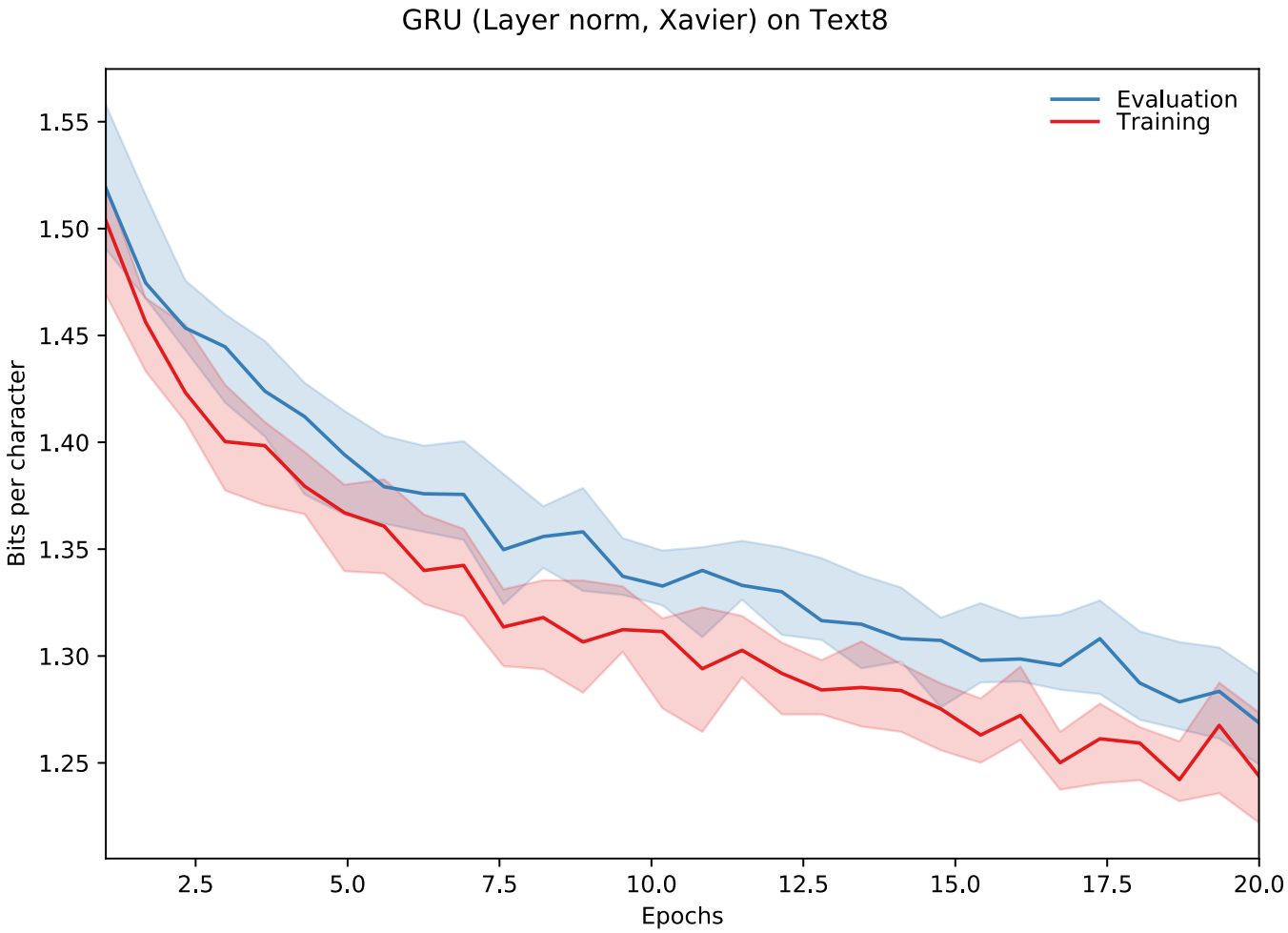
Xavier initialization (Glorot10) samples uniform weights and scales them by the number of incoming and outgoing activations. Variance scaling (He15) is similar, but only considers the incoming activations for scaling and samples from a Gaussian. Orthogonal initialization (Saxe14) is more sophisticated and uses SVD to compute weights that initially preserve the gradient norm.



The choice of initialization did not have a big impact on performance in my experiments. Ironically, the variance scaling initializer resulted in larger variance of the performance. The orthogonal initializer could not show a benefit over the simple Xavier initialization that worked best.

The Best Model

The best model uses GRU memory cells, layer normalization, and the Xavier initializer. With 14 million parameters it gets to 1.27 BPC on the evaluation data, which is better than most reported results on the task (Krueger17, Cooijmans17, Melis17, Zilly17):



In conclusion, try layer norm for your recurrent networks if you haven't, don't worry too much about weight initialization, and consider using GRU, possibly with a larger layer, over LSTM. Feel free to check out my TensorFlow implementation of GRU with Layer norm and Xavier initialization to reproduce the results and for use in your own projects.