Introduction

Previous works only apply batch normalization to the **input-to-hidden** transformation of RNNs, we demonstrate that it is both possible and beneficial to batch-normalize the **hidden-to-hidden** transition, thereby reducing internal covariate shift between time steps.

The Batch-Norm LSTM consistently leads to faster convergence and improved generalization on various sequential problems such as sequence classification, language modeling and question answering.

It's well known that for deep feed-forward neural network, covariate shift degrades the efficiency of training. Covariate shift is a change in the distribution of the inputs to a model. This occurs continuously during training of feed-forward neural networks, where changing the parameters of a layer affects the distribution of the inputs to all layers above it. As a result, the upper layers are continually adapting to the shifting input distribution and unable to learn effectively. The **internal covariate shift** may play an especially important role in recurrent neural networks, which resemble very deep feed-forward networks.

Batch Normalization is a technique for controlling the distributions of feed-forward neural network activations, thereby reducing internal covariate shift. It involves standardizing the activations going into each layer, **enforcing their means and variances to be invariant to changes** in the parameters of underlying layers. This effectively decouples each layer's parameters from those of other layers, leading to a better-conditioned optimizatoin problem. Indeed, dnn trained with bn converge significantly faster and generalize better.

LSTM

It is proven to be difficult to apply in recurrent architectures (Lau-rent et al., 2016; Amodei et al., 2015). It has found limited use in stacked RNNs, where the nor-malization is applied “vertically”, i.e. to the input of each RNN, but not “horizontally” between timesteps.

In what follows, we focus on the LSTM architecture (Hochreiter & Schmidhuber, 1997) with recurrent transition given by

$$\begin{pmatrix} \tilde{\mathbf{f}}_t \\ \tilde{\mathbf{i}}_t \\ \tilde{\mathbf{o}}_t \\ \tilde{\mathbf{g}}_t \end{pmatrix} = \mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{W}_x \mathbf{x}_t + \mathbf{b} \tag{2}$$

$$\mathbf{c}_t = \sigma(\tilde{\mathbf{f}}_t) \odot \mathbf{c}_{t-1} + \sigma(\tilde{\mathbf{i}}_t) \odot \tanh(\tilde{\mathbf{g}}_t) \tag{3}$$

$$\mathbf{h}_t = \sigma(\tilde{\mathbf{o}}_t) \odot \tanh(\mathbf{c}_t), \tag{4}$$

where $\mathbf{W}_h \in \mathbb{R}^{d_h \times 4d_h}$, $\mathbf{W}_x \mathbb{R}^{d_x \times 4d_h}$, $\mathbf{b} \in \mathbb{R}^{4d_h}$ and the initial states $\mathbf{h}_0 \in \mathbb{R}^{d_h}$, $\mathbf{c}_0 \in \mathbb{R}^{d_h}$ are model parameters. σ is the logistic sigmoid function, and the $\odot$ operator denotes the Hadamard product.

The LSTM has an additional memory cell c_t whose update is nearly linear which allows the gradient to flow back through time more easily. the update of the LSTM cell is regulated by a set of gates. The forget gate ft determines the extent to which information is carried over from the previous timestep, and the input gate it controls the flow of information from the current input

xt. **The output gate ot allows the model to read from the cell**. This carefully controlled interaction with the cell is what allows the LSTM to robustly retain information for long periods of time.

Batch Normalization

Batch Normalization (Ioffe & Szegedy, 2015) is a recently proposed network reparameterization which aims to reduce internal covariate shift. It does so by standardizing the activations using empirical estimates of their means and standard deviations.

$$\mathrm{BN}(\mathbf{h};\gamma,\beta) = \beta + \gamma \odot \frac{\mathbf{h} - \widehat{\mathbb{E}}[\mathbf{h}]}{\sqrt{\widehat{\mathrm{Var}}[\mathbf{h}] + \epsilon}} \tag{5}$$

where $\mathbf{h} \in \mathbb{R}^d$ is the vector of (pre)activations to be normalized, $\gamma \in \mathbb{R}^d, \beta \in \mathbb{R}^d$ are model parameters that determine the mean and standard deviation of the normalized activation, and $\epsilon \in \mathbb{R}$ is a regularization hyperparameter. The division should be understood to proceed elementwise.

At training time, the statistics E[h] and Var[h] are estimated by the sample mean and sample variance of **the current minibatch**. This allows for backpropagation through the statistics, preserving the convergence properties of stochastic gradient descent. During inference, the statistics are typically estimated based on the entire training set, so as to produce a deterministic prediction.

Batch-Normalized LSTM

$$\begin{pmatrix} \tilde{\mathbf{f}}_t \\ \tilde{\mathbf{i}}_t \\ \tilde{\mathbf{o}}_t \\ \tilde{\mathbf{g}}_t \end{pmatrix} = \mathrm{BN}(\mathbf{W}_h \mathbf{h}_{t-1}; \gamma_h, \beta_h) + \mathrm{BN}(\mathbf{W}_x \mathbf{x}_t; \gamma_x, \beta_x) + \mathbf{b} \tag{6}$$

$$\mathbf{c}_t = \sigma(\tilde{\mathbf{f}}_t) \odot \mathbf{c}_{t-1} + \sigma(\tilde{\mathbf{i}}_t) \odot \tanh(\tilde{\mathbf{g}}_t) \tag{7}$$

$$\mathbf{h}_t = \sigma(\tilde{\mathbf{o}}_t) \odot \tanh(\mathrm{BN}(\mathbf{c}_t; \gamma_c, \beta_c)) \tag{8}$$

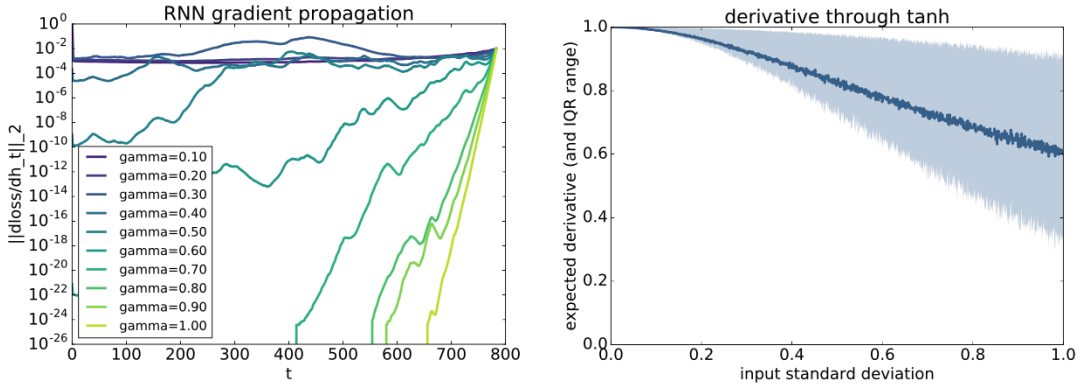
In our formulation, we normalize the recurrent term $\mathbf{W}_h \mathbf{h}_{t-1}$ and the input term $\mathbf{W}_x \mathbf{x}_t$ separately. Normalizing these terms individually gives the model better control over the relative contribution of the terms using the γ_h and γ_x parameters. We set $\beta_h = \beta_x = \mathbf{0}$ to avoid unnecessary redundancy, instead relying on the pre-existing parameter vector $\mathbf{b}$ to account for both biases. In order to leave the LSTM dynamics intact and preserve the gradient flow through $\mathbf{c}_t$, we do not apply batch normalization in the cell update.

The batch normalization transform relies on batch statistics to standardize the LSTM activations. It would seem natural to **share the statistics that are used for normalization across time**, just as recurrent neural networks share their parameters over time. However, we find that simply averaging statistics over time **severely degrades performance**. Although LSTM activations do converge to a stationary distribution, we observe that their statistics during the initial transient differ significantly (see Figure 5 in Appendix A). Consequently, we recommend using separate statistics for each timestep to preserve information of the initial transient phase in the activations.

For our experiments we estimate the population statistics separately for each timestep 1, . . . , Tmax where Tmax is the length of the longest training sequence. When at test time we need to generalize beyond Tmax, we use the population statistic of time Tmax for all time steps beyond it.

During training the author estimate the statistics across the minibatch, independently for each timestep. At test time they use estimates obtained by averaging the minibatch estimates over the training set.

Initializing gammar for Gradient FLow.



(a) We visualize the gradient flow through a batch-normalized tanh RNN as a function of γ. High variance causes vanishing gradient.

(b) We show the empirical expected derivative and interquartile range of tanh nonlinearity as a function of input variance. High variance causes saturation, which decreases the expected derivative.

Figure 1: Influence of pre-activation variance on gradient propagation.

In fig 1.b, when the input standard deviation is low, the input tends to be close to the origin where the derivative is close to 1. As the standard deviation increases, the expected derivative decreases as the input is more likely to be in the saturation regime. At unit standard deviation, the expected derivative is much smaller than 1.

the Authors conjectured that this is what causes the gradient to vanish, and recommend **initializing gamma to a small value**.

Experiments