ON TRAINING RECURRENT NETWORKS WITH TRUNCATED BACKPROPAGATION THROUGH TIME IN SPEECH RECOGNITION

Hao Tang, James Glass

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{haotang,glass}@mit.edu

ABSTRACT

Recurrent neural networks have been the dominant models for many speech and language processing tasks. However, we understand little about the behavior and the class of functions recurrent networks can realize. Moreover, the heuristics used during training complicate the analyses. In this paper, we study recurrent networks' ability to learn long-term dependency in the context of speech recognition. We consider two decoding approaches, online and batch decoding, and show the classes of functions to which the decoding approaches correspond. We then draw a connection between batch decoding and a popular training approach for recurrent networks, truncated backpropagation through time. Changing the decoding approach restricts the amount of past history recurrent networks can use for prediction, allowing us to analyze their ability to remember. Empirically, we utilize long-term dependency in subphonetic states, phonemes, and words, and show how the design decisions, such as the decoding approach, lookahead, context frames, and consecutive prediction, characterize the behavior of recurrent networks. Finally, we draw a connection between Markov processes and vanishing gradients. These results have implications for studying the long-term dependency in speech data and how these properties are learned by recurrent networks.

Index Terms— Recurrent neural networks, speech recognition, Markov processes, backpropagation through time, vanishing gradients

1. INTRODUCTION

Recurrent neural networks (RNNs) have been extensively used for speech and language processing, and are particularly successful at sequence prediction tasks, such as language modeling [1, 2, 3] and automatic speech recognition [4, 5, 6]. Their success is often attributed to their ability to learn long-term dependencies in the data or, more generally, their ability to remember. However, exactly how far back recurrent networks can remember [7, 3], whether there is a limit [8], and how the limit is characterized by the network architecture [9] or by the optimization method [10], are still open questions. In this paper, we examine how training and decoding approaches affect recurrent networks' ability to learn long-term dependencies. In particular, we study the behavior of recurrent networks in the context of speech recognition when their ability to remember is constrained.

The ability to remember in recurrent networks is affected by many factors, such as the amount of long-term dependency in the data, as well as the training and decoding approaches. We differentiate between two types of decoding, the online approach and the batch approach. In the online case, there is a single chain of recurrent network that predicts at all time steps, while in the batch case, multiple chains of recurrent networks that start and end at different time points are used. In the online case, a recurrent network resembles a recursive function where the prediction of the current time step depends on the memory that encodes the entire history. In the batch case, a recurrent network resembles a fixed order Markov process, because the prediction at the current time step strictly depends on a fixed number of time steps in the past. By changing the decoding approach, we restrict the ability of recurrent networks to model certain classes of functions.

The speech signal is particularly rich with long-term dependencies. Due to the causal nature of time, speech is assumed to be Markovian (though with a potentially high order). The order of the Markov property depends on the time scale and the linguistic units, such as subphonetic states, phonemes, and words. The Markov assumption has a strong influence in many design choices, such as the model family or the training approaches. Partly due to the Markov assumption and partly due to computational efficiency [11, 12], recurrent networks in speech recognition [4, 5] and in language modeling [1, 2, 3] are commonly trained with truncated backpropagation through time (BPTT) [13, 14], where a recurrent network is unfolded for a fixed number of time steps for the gradients to propagate. The hypothesis is that even with truncation, recurrent networks are still able to learn recursive functions that are richer than fixed order Markov processes. In fact, recurrent networks at test time are typically applied

to sequences much longer than the ones they are trained on [5, 6]. Under certain conditions, it has be shown that truncation does not affect the performance of RNNs [15]. By our definition, these recurrent networks are trained in batch mode, and are used in an online fashion during testing. We will examine how the number of BPTT steps affects the training loss and how the decoding approaches at test time affect the test loss.

Another factor that impacts the ability of a recurrent network to remember is optimization [16, 10]. Vanilla recurrent networks are known to be difficult to train [17, 10]. Previous work has attributed this difficulty to the vanishing gradient problem [18]. Long short-term memory networks (LSTMs) have been proposed to alleviate the vanishing gradient problem and are assumed to be able to learn longer dependencies in the data [19]. Whether these statements are true is still debatable, but we will draw a connection between the Markov assumption, Lipschitz continuity, and vanishing gradients in recurrent networks. The vanishing gradient phenomenon is in fact a necessary condition for Markov processes.

The contribution of the paper is a comprehensive set of experiments comparing two decoding approaches, various numbers of BPTT steps during training, and their respective results on three types of target units, namely, subphonetic states, phonemes, and words. The results have implications for studying long-term dependency in speech and how well they are learned by recurrent networks.

2. PROBLEM SETTING

We first review the definition of recurrent networks and discuss the types of functions they aim to model.

Let X be the set of input elements, L be the label set, and $Y = \mathbb{R}^{|L|}$ for representing (log-)probabilities or one-hot vectors of elements in L. For example, in the case of speech recognition, the set $X = \mathbb{R}^{40}$ if we use 40-dimensional acoustic features, and L can be the set of subphonetic states, phonemes, or words. Let $\mathcal{X}, \mathcal{Y}$ be the sets of sequences whose elements are in X, Y, respectively. We are interested in finding a function that maps a sequence in $\mathcal{X}$ to a sequence in $\mathcal{Y}$ of the same length. In other words, the goal is to map $(x_1, \ldots, x_T) \in \mathcal{X}$, where each $x_i \in X$, to $(y_1, \ldots, y_T) \in \mathcal{Y}$, where each $y_i \in Y$, for $i = 1, \ldots, T$. We assume we have access to a loss function $\ell : Y \times Y \to \mathbb{R}^+$. For example, the loss function for classification at each time step t can be the cross entropy $\ell(y_t, \hat{y}_t) = -y_t^\top \log \hat{y}_t$, where y_t is the one-hot vector for the ground truth label at time t, and $\log \hat{y}_t$ is a vector of log probabilities produced by the model.

A general recurrent network is a function $s : H \times X \to H$ where H is the space of hidden vectors. For an input sequence $(x_1, \ldots, x_T)$, we repeatedly apply

$$h_t = s(h_{t-1}, x_t) \tag{1}$$

for $t = 1, \ldots, T$ to obtain a sequence of hidden vectors $(h_1, \ldots, h_T)$ where $h_0 = 0$. The hidden vectors are then used for prediction. Note that we are interested in the setting where the length of the input sequence matches the length of the output sequence. Specifically, we have a function $o : H \to Y$ and apply

$$\hat{y}_t = o(h_t) \tag{2}$$

for $t = 1, \ldots, T$ to obtain the label sequence $(\hat{y}_1, \ldots, \hat{y}_T)$. For our purposes, it suffices to use $o(h_t) = \text{softmax}(Wh_t)$ for some trainable parameter matrix W.

A vanilla recurrent neural network implements the above with

$$s(h_{t-1}, x_t) = \sigma(Ux_t + Vh_{t-1}), \tag{3}$$

while a long short-term memory network (LSTM) [19] uses

$$\begin{bmatrix} g_t \\ i_t \\ f_t \\ o_t \end{bmatrix} = \begin{pmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (Ux_t + Vh_{t-1}) \tag{4}$$

$$c_t = i_t \odot g_t + f_t \odot c_{t-1} \tag{5}$$

$$s(h_{t-1}, x_t) = o_t \odot \tanh(c_t) \tag{6}$$

to implement the recurrent network, where $\odot$ is the Hadamard product, σ is the logistic function, $\tanh$ is the hyperbolic tangent, and the matrices U and V are trainable parameters.

Recurrent networks can benefit from stacking on top of each other [20]. We abstract away stacking in the discussion, but will use stacked recurrent networks in the experiments.

2.1. Markov and recursive functions

Recurrent networks can be seen as recursive functions with a constant size memory. To be precise, consider y_t as a function of $x_1, \ldots, x_t$ for any t. We say that y_t is a recursive function if it satisfies $y_t = f(m_t)$ where $m_i = g(m_{i-1}, x_i)$ for $i = 1, \ldots, t$ and any function f and g. The memory is of size constant if the size of m_t is independent of t. In contrast, we say that y_t is a κ-th order Markov function if y_t is a function of $x_{t-\kappa+1}, \ldots, x_t$. By this definition, the set of recursive functions includes all Markov functions, and the inclusion is proper. For example, the sum $y_t = \sum_{i=1}^t x_i$ is not Markov of a fixed order, but it can be realized with a recursive function $y_t = m_t$ where $m_i = m_{i-1} + x_i$ for $i = 1, \ldots, t$. In the language of signal processing, a Markov function resembles a finite-impulse response filter, and a recursive function resembles an infinite-impulse response filter.

2.2. Online and batch decoding

Based on the two function classes, we define two decoding approaches, online and batch decoding, respectively. In online decoding, a sequence of predictions is made one after

another, while in batch decoding, the predictions at different time steps are made independently from each other.

Using the notation in the previous section, to predict $\hat{y}_t$ at time t, we define online decoding as $\hat{y}_t = f(m_t)$ where $m_i = g(m_{i-1}, x_i)$ for $i = 1, \ldots, t$ and any function f and g. During decoding, only the vector m_i at time i needs to be stored in memory, and no history is actually maintained; hence the name online decoding. To implement online decoding with recurrent networks, we simply let $f = o$, $g = s$, and $m_i = h_i$ for all i.

We define batch decoding as $\hat{y}_t = f(x_{t-\kappa+1}, \ldots, x_t)$ for some function f and context size κ. By this definition, batch decoding is not limited to recurrent networks, and can be used with other neural networks. To implement batch decoding with recurrent networks, we let f compute $\hat{y}_t = o(h_t)$, $h_t = z_t^t$, and $z_i^t = s(z_{i-1}^t, x_i)$ where $z_{t-\kappa}^t = 0$, for $i = t - \kappa + 1, \ldots, t$.

In terms of computation graphs, the graph for online decoding with recurrent networks is a single chain, while the graph for batch decoding consists of multiple parallel chains. Note that in batch decoding the hidden vector of each chain starts from the zero vector. In other words, h_i is not a function of h_{i-1} for any i, so the hidden vectors cannot be reused when predicting at different time points. Though batch decoding requires more computation and space, it can be parallelized. Online decoding has to be computed sequentially.

By the above definition, batch decoding with κ context frames aims to realize a κ-th order Markov function. Online decoding, however, aims to realize a recursive function. It has been shown that the number of unrolled steps in vanilla recurrent networks controls the capacity of the model in terms of the Vapnik-Chervonenkis dimension [21]. It is unclear whether recurrent networks truly learn this class of functions. However, by changing decoding approaches, we explicitly restrict the class of functions that recurrent networks can realize.

Finally, one distinct property in speech recognition that is absent in language modeling is the option to look beyond the current time frame. The task of predicting the next word becomes meaningless when the next word is observed. However, it is useful to look a few frames ahead to predict the word while the word is being spoken. Formally, we say that a recurrent network decodes with a lookahead ℓ if $\hat{y}_t = o(h_{t+\ell-1})$. Lookahead can be applied to both online and batch decoding, and it falls back to regular decoding when the lookahead is one frame.

3. BACKPROPAGATION THROUGH TIME

There are many approaches to training recurrent networks [22, 23], and the most successful one by far is backpropagation through time [13]. Backpropagation through time (BPTT) is an approach to compute the gradient of a recurrent network when consuming a sequence. A computation graph is created based on the decoding approaches, and the gradients are propagated back through the computation vertices. It is equivalent to unrolling the recurrent networks for several time steps depending on the decoding approaches; hence the name backpropagation through time.

There are many variants of BPTT with the most popular one being truncated BPTT [14]. In the original definition [14], instead of propagating gradients all the way to the start of the unrolled chain, the accumulation stops after a fixed number of time steps. Training recurrent networks with truncated BPTT can be justified if the truncated chains are enough to learn the target recursive functions. In the modern setting [4], truncated BPTT is treated as regular BPTT with batch decoding, and the number of unrolled steps before truncation is the number of context frames in batch decoding. Henceforth, we will use the term BPTT with batch decoding to avoid confusion.

BPTT with batch decoding is seldom used in practice due to the high computation cost, with the only exception being [4] where the context is set to six frames. A more common approach used in conjunction with BPTT and batch decoding is to predict a batch of frames rather than a single one [5, 12]. Formally, for a single chain of recurrent network to predict p consecutive frames at time t, we have $\hat{y}_{t+i-1} = o(h_{t+\ell+i-2})$ for $i = 1, \ldots, p$ where ℓ is the number of lookahead. Note that this decoding approach is only used for training and is never actually used at test time. Figure 1 summarizes the decoding approaches and the hyperparameters, including lookahead, context frames, and consecutive prediction.

In practice, recurrent networks are trained with a combination of BPTT, batch decoding, lookahead, and consecutive prediction [5, 12]. To speed up training, no frames in an utterance are predicted more than once, and many sequences are processed in batches to better utilize parallel computation on GPUs. In addition, the hidden vectors are sometimes cached [4]. Applying these heuristics, however, creates a mismatch between training and testing. Previous work has not addressed this issue, and the distinction between online and batch decoding under BPTT has only been lightly explored in [4, 11]. We will examine these in detail in our experiments.

4. EXPERIMENTS

To study how well recurrent networks model Markov and recursive functions, we conduct experiments on frame classification tasks with three different linguistic units, i.e., subphonetic states, phonemes, and words. We expect that subphonetic states and phonemes are relatively local in time, while words require longer contexts to predict. By varying the target units, we control the amount of long-term dependency in the data.

Experiments are conducted on the Wall Street Journal data set (`WSJ0` and `WSJ1`), consisting of about 80 hours of read speech and about 20,000 unique words, suitable for studying

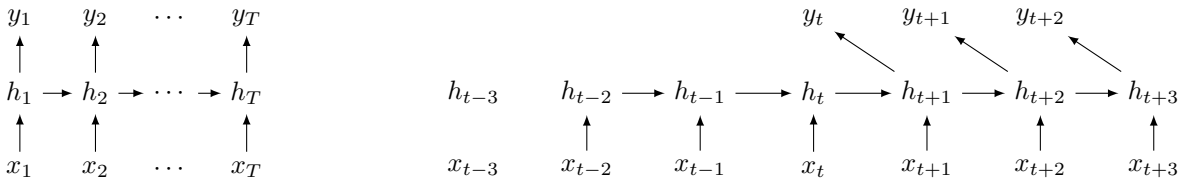

Fig. 1. *Left*: Online decoding for $(x_1, \ldots, x_T)$. *Right*: One chain of batch decoding at time t with 6 context frames, a lookahead of 2 frames, and consecutive prediction of 3 frames. The chains are repeated for $t = 1, \ldots, T$.

rich long-term dependency. We use 90% of `si284` for training, 10% of `si284` for development, and evaluate the models on `dev93`. The set `dev93` is chosen because it is the only set where frame error rates are reported for deep networks [24]. The time-aligned frame labels, including subphonetic states, phonemes, and words, are obtained from speaker-adaptive hidden Markov models following the Kaldi recipe [25].

In the following experiments, we use, as input to the frame classifiers, 80-dimensional log Mel features without appending i-vectors. For recurrent networks, we use 3-layer LSTMs with 512 hidden units in each layer. In addition, for the baseline we use a 7-layer time-delay neural network (TDNN) with 512 hidden units in each layer and an architecture described in [26, 6]. The network parameters are initialized based on [27]. The networks are trained with vanilla stochastic gradient descent with step size 0.05 for 20 epochs. The batch size is one utterance and the gradients are clipped to norm 5. The best performing model is chosen based on the frame error rates (FER) on the development set. Utterances are padded with zeros when lookahead or context frames outside the boundaries is queried.

4.1. Long-term dependency in subphonetic states

We first experiment with recurrent networks on subphonetic states. We expect LSTMs to perform best when trained and tested with online decoding, so we first explore the effect of lookahead. Results are shown in Table 1. Lookahead has a significant effect on the frame error rates. The error rate improves as we increase the amount of lookahead and plateaus after 10 frames. The improvement is due to better training error (not regularization or other factors), as shown in Figure 2. In addition, the fact that increasing the amount of lookahead does not hurt performance suggests that LSTMs are able to retain information for at least 20 frames. Compared to prior work, our 3-layer LSTM is unidirectional and uses fewer layers than the ones used in [24], but the results are on par with theirs. The best LSTM achieves 11.7% word error rate on `dev93`, similar to the results in [24].[1] The TDNN serves as an instance of a Markov function. We present its result, but more investigations are needed to conclude anything from it.

[1]We are aware of the state of the art on this data set [28]. As the results are in the ballpark, we do not optimize them further.

Table 1. FERs (%) comparing different LSTM lookaheads trained with online decoding, on subphonetic states. The FER of a 7-layer TDNN is provided as a reference.

lookahead	dev	`dev93`
1	45.46	43.45
5	33.64	32.43
10	30.43	29.83
15	29.89	28.69
20	29.36	28.59
TDNN 512x7	33.56	34.19

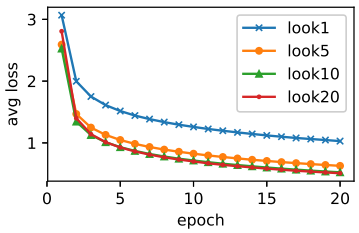

Fig. 2. Running average of training losses for comparing different amount of lookahead for LSTMs with online decoding trained on subphonetic states.

To see how decoding approaches affect performance, we examine the error rates with batch decoding on the best LSTM in Table 1, i.e., the one trained with online decoding and a lookahead of 20 frames. Results are shown in Table 2. The performance deteriorates as we reduce the number of context frames. This suggests that LSTMs do utilize information 40 frames away. On the other hand, the degradation is not severe, suggesting that much of the long-term dependency is Markov.

We then examine whether LSTMs with batch decoding can learn recursive functions. The same LSTMs are trained with batch decoding of different context frames. Consecutive prediction is not used in this setting. In other words, for an utterance of T frames, we create T chains of LSTMs, each of which predicts the label of a frame. Results are shown in Table 3. With a context of 40 frames, the LSTM trained with batch decoding is only slightly behind the LSTM trained with online decoding. This again suggests that the class of Markov

Table 2. FERs (%) comparing different context frames under batch decoding with the best LSTM trained with online decoding, and a lookahead of 20 frames on subphonetic states.

context	dev	dev93
40	37.65	36.45
35	41.73	40.12
30	48.79	46.70
online	29.36	28.59

Table 3. FERs (%) on dev93 comparing online and batch decoding for LSTMs trained with batch decoding, and a lookahead of 20 frames on subphonetic states.

context	batch	online
40	31.30	80.29
35	30.99	84.78
30	32.80	85.74

functions can perform reasonably well, and much of the long-term dependency is likely Markov. However, the error rate degrades significantly when we switch from batch decoding to online decoding. This strongly suggests that these LSTMs do not behave like recursive functions.

To understand what makes LSTMs behave like recursive functions, we train LSTMs with batch decoding and increase the amount of consecutive prediction. To simplify the setting, we allow each frame to be predicted multiple times. In fact, if a network predicts consecutively for p frames, then each frame gets predicted p times. Results are shown in Table 4. As we increase the number of consecutively predicted frames, the error rate for batch decoding stays about the same, while the one for online decoding improves. This suggests that it is the amount of consecutive prediction that gears the behavior of LSTMs towards recursive functions. In addition, as seen in Table 2, LSTMs can achieve reasonable performance with both online and batch decoding. Perhaps the data is a complex mix of long-term dependency, or perhaps the LSTM learns to forget the history. More analyses are needed to tease the factors apart.

4.2. Long-term dependency in phonemes and words

We repeat the same experiments with phonemes and words. Note that the number of classes in the case of words is the number of unique words in the training set. We do not handle out-of-vocabulary words (OOV).

For the lookahead experiments in Table 5, the general trend is similar to that in Table 1, except that the frame error rate plateaus at around 15 frames for words, longer than the 10 frames in subphonetic states and phonemes.

The fact that the TDNN achieves a low frame error rate for phonemes in Table 5 suggests that much phonetic information is concentrated within a 30-frame window. However, for words we expect a larger context window to predict words, so we evaluate a 10-layer TDNN with an effective context window of 48 frames. The 10-layer TDNN performs better than the 7-layer one, but is still behind LSTMs.

Table 4. FERs (%) on dev93 comparing different numbers of consecutive prediction for LSTMs trained with batch decoding, and a lookahead of 20 frames on subphonetic states.

# of prediction	batch	online
1	31.30	80.29
5	31.29	80.21
10	31.65	46.82
15	32.27	33.03

Table 5. FERs (%) comparing different lookaheads for LSTMs trained with online decoding, on phonemes and words. The FERs of a 7-layer TDNN and a 10-layer TDNN are provided as reference.

	phonemes		words	
lookahead	dev	dev93	dev	dev93
1	18.31	16.34	32.78	45.60
5	13.57	12.01	28.39	40.36
10	12.33	11.01	24.68	36.62
15	11.90	10.67	21.57	32.82
20	11.86	10.76	20.85	31.45
25			21.02	30.43
TDNN 512x7	14.22	12.46	39.27	38.93
TDNN 512x10			28.97	32.24

The conclusion in Table 6 is the same as in Table 2, i.e., for LSTMs trained with online decoding, reducing the amount of context during batch decoding hurts the accuracy. However, the conclusion in Table 7 is different from that in Table 3. The LSTMs with batch decoding are able to perform well with online decoding on phonemes but not on words. We suspect that phonemes are more local than subphonetic states, and in general the results are affected by the choice of linguistic units.

In Table 8, we observe a similar trend to that in Table 4, adding consecutive prediction improves the LSTMs' performance with online decoding. In fact, the performance with online decoding, compared to that in Table 5, is fully recovered when using 15 frames of consecutive prediction.

5. MARKOV ASSUMPTION, LIPSCHITZ CONTINUITY, AND VANISHING GRADIENTS

From the experiments, it is difficult to conclude whether the LSTMs really learn Markov or recursive functions. In this section, we derive a necessary condition that we can check empirically for Markov functions. A function f is

Table 6. FERs (%) for batch decoding with different context frames with the best LSTM trained with online decoding, and a lookahead of 20 frames on phonemes and words.

	phonemes		words	
context	dev	dev93	dev	dev93
40	16.74	14.99	48.39	44.84
35	19.28	17.10	53.33	48.55
30	23.26	20.59	58.78	53.00
online	11.86	10.76	20.85	31.45

Table 7. FERs (%) on dev93 comparing online and batch decoding for LSTMs trained with batch decoding, and a lookahead of 20 frames on phonemes and words.

	phonemes		words	
context	batch	online	batch	online
40	11.50	11.24	31.65	56.96
35	12.12	15.11	34.21	58.26
30	12.89	21.97	37.83	64.74

G-Lipschitz with respect to i-th coordinate and norm $\|\cdot\|$ if

$$\Big| f(a_1, \ldots, a_{i-1}, a, a_{i+1}, \ldots, a_t) - f(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_t) \Big| \leq G\|a - b\| \quad (7)$$

for any $a_1, \ldots, a_t$ and any a, b. In words, the function does not change too much when we perturb the i-th coordinate. By definition, a Markov function does not change at all when we perturb the coordinate beyond the necessary history, or more formally, a κ-th order Markov function is 0-Lipschitz with respect to $x_1, \ldots, x_{t-\kappa}$. The Lipschitz property relates to the gradient through the perturbation interpretation. A gradient of a function can be regarded as having an infinitesimal perturbation of the coordinates, so the i-th coordinate of a gradient has a value at most G for a G-Lipschitz function with respect to the i-th coordinate. A κ-th order Markov function should have zero gradient with respect to $x_1, \ldots, x_{t-\kappa}$, because it is 0-Lipschitz with respect to those coordinates. In other words, the gradients to the input, and thus to the parameters, must vanish after κ steps for a κ-th order Markov function.

In practice, partly because of noise and partly because we do not know if the data is really Markov, it is better not to enforce vanishing gradients in the model. Regardless, this gives a necessary condition we can check empirically. We take the best online and batch LSTMs trained on subphonetic states from Tables 1 and 3. The norms of gradients to the input at time $t - 20$ when predicting at time t is shown in Figure 3. The norms of the LSTM trained with batch decoding concentrate near zero compared to the ones with online decoding. This confirms that LSTMs trained with batch decoding behave like Markov functions.

Table 8. FERs (%) on dev93 comparing different numbers of consecutive prediction for LSTMs trained with batch decoding, and a lookahead of 20 frames on phonemes and words.

	phonemes		words	
# of prediction	batch	online	batch	online
1	11.50	11.24	31.65	56.96
5	11.58	11.68	31.79	45.88
10	11.49	10.89	32.31	36.40
15	11.48	10.72	32.34	30.93

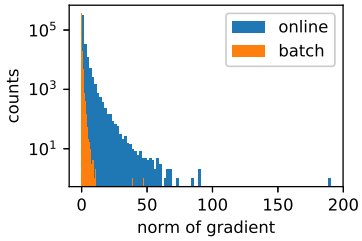

Fig. 3. The norms of gradients to the input x_{t-20} when predicting y_t on dev93 comparing LSTMs trained with online and batch decoding with 40 context frames. Both LSTMs are trained on subphonetic states with a lookahead of 20 frames. Note that the counts are in log scale.

6. CONCLUSION

We study unidirectional recurrent networks, LSTMs in particular, trained with two decoding approaches, online and batch decoding, and explore various hyperparameters, such as lookahead, context frames, and consecutive prediction. Online decoding can be as broad as recursive functions, while batch decoding can only include Markov functions. Training LSTMs with the matching decoding approaches performs best. LSTMs trained with online decoding can still have decent performance with batch decoding, but LSTMs trained with batch decoding tend not to perform well with online decoding. The amount of lookahead is also critical for LSTMs with online decoding to get better training errors. The number of context frames strictly limits the history that can be used for prediction, while increasing the amount of consecutive prediction gears LSTMs closer to recursive functions. The results depend on the long-term dependency in the data, and we confirm this by exploring subphonetic states, phonemes, and words. Finally, we show that the vanishing gradient phenomenon is a necessary condition for Markov functions with respect to variables beyond the necessary history. It is important to note that the same LSTM architecture and the same number of model parameters can have drastically different results and behaviors. We hope the results improve the understanding of recurrent networks, what they learn from the data, and ultimately the understanding of speech data itself.

7. REFERENCES

[1] Stephen Merity, Nitish Shirish Keskar, and Richard Socher, "Regularizing and optimizing LSTM language models," in *International Conference on Learning Representations (ICLR)*, 2018.

[2] Stephen Merity, Nitish Shirish Keskar, and Richard Socher, "An analysis of neural language modeling at multiple scales," *arXiv:1803.08240*, 2018.

[3] Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky, "Sharp nearby, fuzzy far away: How neural language models use context," in *Annual Meetings of the Association for Computational Linguistics (ACL)*, 2018.

[4] George Saon, Hagen Soltau, Ahmad Enami, and Michael Picheny, "Unfolded recurrent neural networks for speech recognition," in *Interspeech*, 2014.

[5] Haşim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech*, 2014.

[6] Vijayaditya Peddinti, Yiming Wang, Daniel Povey, and Sanjeev Khudanpur, "Low latency acoustic modeling using temporal convolution and LSTMs," *IEEE Signal Processing Letters*, vol. 25, 2018.

[7] Olivia L. White, Daniel D. Lee, and Haim Sompolinsky, "Short-term memory in orthogonal neural networks," *Physical Review Letters*, vol. 92, 2004.

[8] Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo, "Capacity and trainability in recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[9] Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Ruslan R. Salakhutdinov, and Yoshua Bengio, "Architectural complexity measures of recurrent neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[10] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural netwokrs," in *International Conference on Machine Learning*, 2013.

[11] Naoyuki Kanda, Mitsuyoshi Tachimori, Xugang Lu, and Hisashi Kawai, "Training data pseudo-shuffling and direct decoding framework for recurrent neural network based acoustic modeling," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015.

[12] Kai Chen, Zhi-Jie Yan, and Qiang Huo, "Training deep bidirectional LSTM acoustic model for LVCSR by a context-sentisitve-chunk BPTT approach," in *Interspeech*, 2015.

[13] Paul J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, 1990.

[14] Ronald J. Williams and Jing Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Computation*, 1990.

[15] John Miller and Moritz Hardt, "When recurrent models don't need to be recurrent," 2018.

[16] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, 1994.

[17] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and J urgen Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Networks*. Wiley-IEEE Press, 2001.

[18] Sepp Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, 1998.

[19] Sepp Hochreiter and J urgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, 1997.

[20] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[21] Pascal Koiran and Eduardo D. Sontag, "Vapnik-Chervonenkis dimension of recurrent neural networks," Tech. Rep. 96-56, DIMACS, 1996.

[22] Luis B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *IEEE International Conference on Neural Networks*, 1987.

[23] Fernando J. Pineda, "Generalization of backpropagation to recurrent neural networks," *Physical Review Letters*, vol. 59, 1987.

[24] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2013.

[25] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý, "The Kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[26] Hao Tang, Wei-Ning Hsu, François Grondin, and James Glass, "A study of enhancement, augmentation, and autoencoder methods for domain adaptation in distant speech recognition," in *Interspeech*, 2018.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *International Conference on Computer Vision (ICCV)*, 2015.

[28] William Chan and Ian Lane, "Deep recurrent neural networks for acoustic modeling," *arXiv:1504.01482*, 2015.