SAMPLED CONNECTIONIST TEMPORAL CLASSIFICATION

Ehsan Variani[1], *Tom Bagby*[1], *Kamel Lahouel*[2], *Erik McDermott*[1], *Michiel Bacchiani*[1]

[1]Google Inc., USA
[2]Johns Hopkins Univ., Baltimore, MD USA

{variani,tombagby,erikmcd,michiel}@google.com klahouel@jhu.edu

ABSTRACT

This article introduces and evaluates Sampled Connectionist Temporal Classification (CTC) which connects the CTC criterion to the Cross Entropy (CE) objective through sampling. Instead of computing the logarithm of the sum of the alignment path likelihoods, at each training step the sampled CTC only computes the CE loss between the sampled alignment path and model posteriors. It is shown that the sampled CTC objective is an unbiased estimator of an upper bound for the CTC loss, thus minimization of the sampled CTC is equivalent to the minimization of the upper bound of the CTC objective. The definition of the sampled CTC objective has the advantage that it is scalable computationally to the massive datasets using accelerated computation machines. The sampled CTC is compared with CTC in two large-scale speech recognition tasks and it is shown that sampled CTC can achieve similar WER performance of the best CTC baseline in about one fourth of the training time of the CTC baseline.

Index Terms— Deep Neural Networks, CTC, Cross Entropy, Sampling.

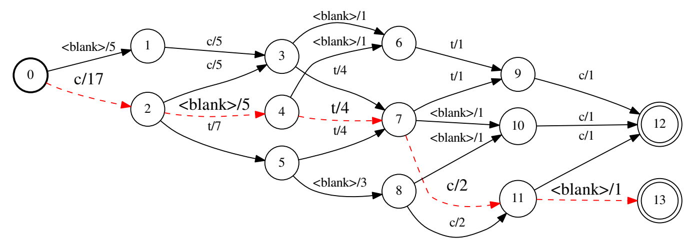

Fig. 1. The path inventory graph for ground truth alignment `|c t t t c|`. There is total of 22 alignment paths in this graph. Each path is corresponding to an alignment which outputs the ground truth labels within at most one frame of the alignment boundaries. The edge's weight is the total number of alignment paths starting from that edge which ends in one of the final states. The dashed red path is a sample path uniformly selected based on the count distributions.

1. INTRODUCTION

Connectionist Temporal Classification (CTC) [1] is a sequence training criterion defined as the posterior of observation sequence given feature sequence. The criterion is computed by marginalizing the posterior of all possible alignments which contain an additional blank symbol in between ground truth label sequence. To deal with the exponential number of possible alignments, the summation is performed via forward-backward algorithm [2]. When integrated with deep architectures, CTC imposes specific memory and computation requirements which can be a bottleneck for training scalability on accelerated machines like Graphics Processing Units (GPUs) [3, 4].

In speech recognition, CTC loss often deployed for training deep Recurrent Neural Networks (RNNs) with Long Short Term Memory (LSTM) cells [5, 6, 7, 8, 9] or Gated Recurrent Units (GRUs) [10, 3]. Training RNN with a sequence losses like CTC requires full unrolling of RNN by input sequence length which imposes extra memory requirements and limits the scalability of training parallelization on GPUs through batching [4, 11]. For practical speech recognition systems, the training corpus usually contains utterances with variable length which makes parallel training with multiple sequences in a minibatch inefficient both in terms of memory consumption and computation. Furthermore, the forward-backward algorithm has $\mathcal{O}(N^2T)$ computation and memory complexity [2] where T and N are the length of input sequence and number of softmax classes, respectively. For large scale Automatic Speech Recognition (ASR) systems, the number of classes is order of thousands and average utterance length is of the order of hundreds. These computation and memory constraints significantly add to the complexity of training a RNN architecture with time back-propagation. To overcome the computation complexity of CTC, the general practise is either efficient machine specific implementation of the forward-backward algorithm [3, 12, 13] or implementation of some approximation of the CTC objective [4]. However, the CTC loss computation is still computationally much more expensive than frame-level objectives like Cross Entropy (CE) which can be easily scaled using truncated unrolling and large batches on GPUs [11].

This paper connects the sequence-level CTC criterion to the frame-level CE training through sampling. Instead of computing CTC loss by summing over all possible alignments containing blank symbol, we investigate relabeling an input sequence by a label sequence randomly chosen from all possible alignments allowed by the alignment inventory. Given the chosen alignment, the CE loss between softmax posteriors and selected alignment as well as backward gradients are computed to update the deep architecture weights. The mathematical connection of the proposed objective and CTC loss is explained. Two sampling methods are presented for constructing a uniform alignment path inventory given the ground truth alignment. Finally, the proposed method is experimentally evaluated in two Large Vocabulary Continuous Speech Recognition (LVCSR) tasks. It is shown that the sampled CTC method allow scalability of training time while preserving similar quality performance yield by training with CTC objective.

2. SAMPLED CTC

2.1. Connectionist Temporal Classification (CTC)

For an acoustic feature sequence $\bar{x} = x_1, ..., x_T$ and word transcript $\bar{w} = w_1, ..., w_M$, the CTC criterion is defined as

$$\begin{aligned} L_{CTC}(\theta, \bar{w}, \bar{x}) &:= -\log\left(\sum_{\bar{s}\in\mathcal{S}_{\bar{w}}} p_\theta(\bar{s}|\bar{x})\right) \\ &= -\log\left(\sum_{\bar{s}\in\mathcal{S}_{\bar{w}}} \prod_{t=1}^{T} p_\theta(s_t|x_t)\right) \end{aligned} \quad (1)$$

where θ denotes the model parameters, $\mathcal{S}_{\bar{w}}$ is the path inventory containing all the alignment paths like $\bar{s} = s_1, ..., s_T$ allowed by the CTC topology. Here s_t for each time step t can be either a class symbol or a blank symbol. For a given alignment, a valid path in the CTC topology is a path with edit distance of 0 to the ground truth label sequence after removal of all the blank symbols and consecutive repetitions. For example, if the sequence of symbols from ground truth alignment is `|c t t t c|`, the resulting CTC path inventory should encodes all the paths with the following pattern: `|<blank>* c+ <blank>* t+ <blank>* c+ <blank>*|` where `<blank>` denotes the blank symbol, $*$ means 0 or more repetitions, and $+$ means 1 or more repetitions.

The CTC objective is usually computed using Finite State Transducers (FSTs) by composition of a *score* transducer S and *label* transducer L [6]. The score transducer encodes state s_t and its posterior $p(s_t|x_t)$ for each time step t. The label transducer encodes all possible paths allowed by the CTC topology. It is built by composition of the string transducer for the ground truth alignment with a transducer which converts ground truth alignment to sequence of states with repetitions interleaved with optionally blank symbols. Figure 1 shows a path inventory representation of L when ground truth alignment is `|c t t t c|`. The forward loss and backward gradient computation is performed using shortest path algorithm on $S \circ L$ fst.

The CTC topology imposes no alignment constraints on the paths, which in turn enables any state symbol to occur at anytime. It is likely that the spiky posteriors occur towards the end of the utterance which is not desirable feature for practical streaming recognition systems like voice-search [7]. To fulfill the streaming requirements, instead of building an *unconstrained* label transducer which encodes all possible paths allowed by the CTC topology, a *constrained* transducer is built which limits the valid paths to those in which the delay between the CTC labels and the ground truth alignment does not exceed some threshold. For example, the label path inventory of Figure 1 allows delay threshold of 1 which let the paths `|<blank> c t t c|` or `|c t <blank> c <blank>|` be permitted while not allowing paths like `|<blank> <blank> <blank> c t c|`.

2.2. Sampled CTC

The CTC objective of Eq 1 can be bounded by expected value of the path log-likelihoods under some prior knowledge of the paths distribution. Assuming the alignment paths in the state inventory $\mathcal{S}_{\bar{w}}$ be distributed by a distribution $q(\bar{s})$ which assigns non zero distribution mass to each alignment path $\bar{s}$ in the path inventory $\mathcal{S}_{\bar{w}}$:

$$\begin{aligned} L_{CTC}(\theta, \bar{w}, \bar{x}) &= -\log\left(\sum_{\bar{s}\in\mathcal{S}_{\bar{w}}} \frac{p_\theta(\bar{s}|\bar{x})}{q(\bar{s})}\, q(\bar{s})\right) \\ &\leq -\sum_{\bar{s}\in\mathcal{S}_{\bar{w}}} q(\bar{s}) \log\left(\frac{p_\theta(\bar{s}|\bar{x})}{q(\bar{s})}\right) \\ &= \mathbb{E}_{\mathbf{\bar{s}}}\left[-\log\left(\frac{\mathbf{p}_\theta(\mathbf{\bar{s}}|\bar{x})}{q(\mathbf{\bar{s}})}\right)\right] \end{aligned} \quad (2)$$

where the inequality holds due to the Jensen's inequality [14]. In the final equation, $\mathbf{p}(\mathbf{\bar{s}}|\bar{x})$ is a random variable which is a function of the paths random variable $\mathbf{\bar{s}}$ distributed by $q(\mathbf{\bar{s}})$. The above equation provides an upper bound for the CTC objective which means instead of minimizing the CTC criterion directly, one can minimize the upper bound expected log-likelihood. Furthermore, we argue that minimization of the above expected value is equal to minimization of the log-likelihood directly. Let us define the following objective function:

$$L(\theta, \mathbf{\bar{s}}, \bar{w}, \bar{x}) = -\log\left(\mathbf{p}_\theta(\mathbf{\bar{s}}|\bar{x})/q(\mathbf{\bar{s}})\right) \quad (3)$$

For some acoustic feature sequence $\bar{x}$ and corresponding word sequence $\bar{w}$, the update rule for the gradient descent optimization of the above log-likelihood loss at step n is:

$$\theta_{n+1} = \theta_n - \gamma \frac{\partial}{\partial\theta} L(\theta, \mathbf{\bar{s}}, \bar{w}, \bar{x}) \quad (4)$$

where γ is the learning rate. Taking expected value of the above equation with respect to the path random variable $\mathbf{\bar{s}}$:

$$\begin{aligned} \mathbb{E}_{\mathbf{\bar{s}}}[\theta_{n+1}] &= \mathbb{E}_{\mathbf{\bar{s}}}\left[\theta_n - \gamma \frac{\partial}{\partial\theta} L(\theta, \mathbf{\bar{s}}, \bar{w}, \bar{x})\right] \\ &= \theta_n - \gamma \mathbb{E}_{\mathbf{\bar{s}}}\left[\frac{\partial}{\partial\theta} L(\theta, \mathbf{\bar{s}}, \bar{w}, \bar{x})\right] \\ &= \theta_n - \gamma \frac{\partial}{\partial\theta} \mathbb{E}_{\mathbf{\bar{s}}}\left[L(\theta, \mathbf{\bar{s}}, \bar{w}, \bar{x})\right] \end{aligned} \quad (5)$$

where the first equality is due to the fact that uniform distribution over paths does not depend on the model parameters θ and the second equality holds since the expected value and partial derivative operation are interchangeable. In fact, the expected value in this situation is just a sum over a finite set. Hence, the derivative of path log-likelihood is an unbiased estimator of the derivative of expected value of path log-likelihood. The final right hand side equation is the update rule of model trained with $\mathbb{E}_{\mathbf{\bar{s}}}\left[-\log\left(\frac{\mathbf{p}_\theta(\mathbf{\bar{s}}|\bar{x})}{q(\mathbf{\bar{s}})}\right)\right]$ objective at step n assuming initialized from same model parameter θ_n. Thus model optimized with the path log-likelihood objective of Eq 3 is unbiased estimator of the upper bound of the CTC criterion in Eq 2.

We notate the criterion of Eq 3 as sampled CTC objectives. In lack of prior knowledge about the distribution of the paths inventory, we can assume all the paths are equally likely. In addition, assuming conditional independence like CTC in Eq 1, the sampled CTC loss becomes:

$$L(\theta, \mathbf{\bar{s}}, \bar{w}, \bar{x})\,|_{\mathbf{\bar{s}} \sim U(\mathcal{S})} = -\sum_{t=1}^{T} \log\left(p_\theta(s_t|x_t)\right) \quad (6)$$

which is the cross entropy objective between model posterior and sampled alignment with additional blank symbol class. The training process works as follow. For each utterance with given ground truth alignment, the alignment path inventory $\mathcal{S}_{\bar{w}}$ is first built. According to the construction, the path inventory can be distributed under some prior distribution $q\left(\mathcal{S}_{\bar{w}}\right)$. At each training step, the input sequence is relabeled by an alignment path randomly chosen from $\mathcal{S}_{\bar{w}}$ according to prior $q\left(\mathcal{S}_{\bar{w}}\right)$. Training proceeds by computing the forward cross entropy object between model posteriors and the randomly selected alignment. Note that an utterance can be labeled with different alignment paths as training continues.

The choice of uniform prior distribution over paths inventory is not necessarily the most desirable option. It is clear that as training proceeds, the alignments distribution is changing and model might prefers some alignments over others; indeed if $q\left(\bar{s}\right)$ be proportional to $p_\theta(\bar{s}|\bar{x})$ for all $\bar{s} \in \mathcal{S}_{\bar{w}}$, the equality in Eq 2 holds. Since the main objective of the sampled CTC method is acquiring computation efficiency over the forward-backward algorithm, the uniform distribution is a suitable choice to parallelize the training procedure. This allows Sampled CTC objective be computed without construction of the $S \circ L$ transducer or forward-backward computation on the full sequence posteriors. As shown later, the sampled CTC can be optimized by full unrolling as well as truncated unrolling with large batches which can significantly speed up training. Next we present two ways of constructing a uniform path inventory.

Coin Flipping: Given the ground truth alignment sequence of length T, at each time position t, the path inventory is constructed by flipping an unbiased coin. According to the flip outcome, the ground truth alignment is either kept as is or changed to a blank symbol. Since the coin is unbiased, this simple algorithm constructs an alignment path inventory of size 2^T where all the paths are equally likely (in terms of being blank or not). The path inventory constructed with Coin Flipping is larger than the path inventory of the CTC topology. Unlike CTC path inventory, there is no pattern imposed for valid alignment paths. For the ground truth alignment of `|c t t t c|`, the Coin Flipping inventory allows alignments like `|c <blank> c t c|` or `|c t <blank> t c|` which are not allowed by the CTC topology. The Coin Flipping however is so simple and does not even require to be constructed in advance since the random path can be selected on-the-fly.

Path Counting: Here the path inventory is constructed exactly similar to the CTC path inventory. Figure 1 presents the path inventory for the ground truth alignment of `|c t t t c|`. For each edge in this graph, the total number of paths started from the edge and ends in one of the final states is counted and assigned to the edge's weight. For example in Figure 1 there is total of 17 paths starting from edge $\{0 \rightarrow 2\}$ and there are only 4 paths containing edge $\{4 \rightarrow 6\}$ which started from state 4. To uniformly sample a path from this path inventory, starting from start state 0 we select next edge according to the distribution of the edge's weights which corresponds to the total paths leaving from current state. The path extends by the randomly chosen edge and this process continues until the alignment path reaches one of the final states.

Figure 1 shows a sample path with red dashed edges. At state 0, the possibility of choosing edge $\{0 \rightarrow 1\}$ or edge $\{0 \rightarrow 2\}$ is distributed with $\{\frac{5}{22}, \frac{17}{22}\}$. The randomly selected edge in this example is the edge $\{0 \rightarrow 2\}$. The path grows by making another random selection among the three possible edges $\{2 \rightarrow 3\}, \{2 \rightarrow 4\}, \{2 \rightarrow 5\}$ which are distributed by $\{\frac{5}{17}, \frac{5}{17}, \frac{7}{17}\}$. This process continues until reaching one of the two final states which is state 13 in this example. The selected path in this example is: $\{0 \rightarrow 2\}\{2 \rightarrow 4\}\{4 \rightarrow 7\}\{7 \rightarrow 11\}\{11 \rightarrow 13\}$ which is corresponding to `|c <blank> t c <blank>|` alignment. The probability of this path is: $\frac{17}{22} \times \frac{5}{17} \times \frac{4}{5} \times \frac{2}{4} \times \frac{1}{2}$ which is $1/22$.

3. EXPERIMENTS

The main advantage of the sampled CTC over CTC is argued to be the computation scalability. Unlike CTC objective, the sampled CTC criterion does not necessarily require full unrolling of the recurrent architecture and can be optimized with truncated unrolling and large batches. The experiments are conducted to evaluate the performance of the proposed sampled CTC methods in both full and truncated unrolling setting. Different training processes are compared in terms of the WER performance and computation efficiency.

Data: The training data consists of about $20,000$ hours of spontaneous speech from anonymized, human-transcribed Voice Search queries. For noise robustness training, each utterance in the training set is artificially noisified using 100 different styles of noises with varying degrees of noise and reverberation. The test sets are separate anonymized, human-transcribed Voice Search datasets of about 25 hours each. Evaluation is presented on two sets, *voice-search* which contains short queries from real Google voice search traffic and *dictation*, which contains relatively longer queries. The average number of frames after endpointing is 114 with standard deviation of 296. The dataset is forced-aligned with state inventory of 8192 states.

Front-end: The training examples are 512-dimensional frames output by the front end every 30 ms, generated by stacking four frames of 128-dimensional log mel filterbank energies extracted from a 32 ms window shifted over the audio samples in 10 ms increments [15]. Each frame is labeled with one out of 8192 context-dependent output phoneme states. The feature extraction was performed on-the-fly as explained in [11]. Same front-end was used for all experiments presented here.

Training: For all experiments, a stack of 5 LSTM layers with 768 cells per layer is used which lead to model of 22M parameters. ASGD was used for all experiments. The CPU training was performed with 500 worker and 99 parameter server machines. The GPU experiments used 32 K40 GPUs with 7 parameter servers. The number of machines were optimized for each task to achieve the highest number of training steps per second while reaching the best WER performance. Same model topology has been used for all experiments. The training was conducted in TensorFlow [16] using the system described in [11]. All models are decoded with a two-pass WFST-based decoder. The decoding resources for the CTC model was used for decoding the sampled CTC models. The lm weight and blank scale were optimized separately for each model.

Baseline: Our best CTC setup is trained with full unrolling and batch size of 1 on CPU. The learning rate of $1.5e-4$ with exponential decay rate of 0.9 and decay step of $1e+9$ was used for optimization. The WER performance of this setup on the voice-search and dictation tasks are shown in Table 1. The model is trained for 60 epochs with approximately 1 epoch per day. The performance seems getting better and better as training proceeds however it is relatively slow. Training can be speed up using batching and full unrolling on GPU, however none of our experimental setups were performing as good as the batch one setup of Table 1. We observed batching of the full unrolling setup with CTC loss is very sensitive on the batching parameters. The training data used here varies largely in terms of the utterance length which makes it hard to find the optimum batching parameters which improves training time while not hurting WER performance.

WER [%]	epoch 10	epoch 20	epoch 30	epoch 60
voice-search	10.6	10.3	10.0	9.6
dictation	9.2	8.7	8.5	8.0
training [days]	10	20	31	60

Table 1. WER for the baseline CTC system trained for 60 epochs.

Sampled CTC: This experiment compares CTC and sampled CTC when all the training parameters are identical. The exact CPU CTC setup with full unrolling and batch size of 1 was used here. The optimization parameters like learning rate and exponential decay parameters are all set as in the baseline CTC setup. Table 2 compares the performance of CTC and sampled CTC at epoch 10. Both sampling CTC approaches are performing relatively close to the baseline CTC, though the Coin Flipping method seems little behind the CTC baseline. But more than WER performance, maybe the most promising observation in Table 2 is that even for full unrolling, the training time of sampled CTC is about half of the CTC training time. The CTC criterion needs to perform forward-backward at each step to calculate loss and backward derivatives while the sampled CTC is as cheap as CE training at each step.

WER [%]	CTC	sampling CTC	
		Coin Flipping	Path Counting
voice-search	10.6	11.0	10.6
dictation	9.2	9.8	9.4
training [days]	10	6	6

Table 2. CTC vs. sampled CTC at epoch 10 (full unrolling).

Truncated Unrolling: One of the main advantages of the sampled CTC over CTC is that the sampled CTC can be trained with truncated unrolling and large batches. To examine the behavior of the Sampled CTC in truncated unrolling setup, the both proposed methods were compared on GPU setup with the batch size of 256 and unrolling window of 20 frames. All setups use same initial learning rate and optimization parameters.

Table 3 presents the performance of Coin Flipping model. As training proceeds, the gap between sampled CTC with Coin Flipping the best CTC baseline performances in Table 1 is getting smaller and smaller but the Coin Flipping is lagging behind the baseline CTC even after 40 epochs of training. Another observation is that while training with truncated unrolling is faster in terms of number of epochs per day compared to the full unrolling training of Table 2, the performance of the full unrolling setup at epoch 10 is better than the performance of truncated setup at same epoch. So part of the differences between sampled CTC performance and CTC performance in terms of number of training epochs might be just because of the differences between full unrolling and truncated unrolling.

Table 4 presents the WER performance as well as training time of the sampled CTC with Path Counting approach with truncated unrolling. The batch size and unrolling window of the previous experiment has been used here. The Counting Path methods at epoch 160 of truncated unrolling matching the performance of the CTC setup at epoch 60 in Table 1. The training time for the sampled CTC to reach the same performance of the CTC baseline is about one fourth of the training time of the CTC model. Similar to the Coin Flipping

WER [%]	epoch 10	epoch 20	epoch 40
voice-search	11.5	11.8	10.9
dictation	10.4	10.0	9.3
training [days]	1.5	3.6	7.2

Table 3. Sampled CTC with Coin Flipping (truncated unrolling).

experiment, the Path Counting with truncated unrolling at epoch 10 performs worst than same epoch with full unrolling in Table 2.

The fact that sampled CTC can be trained with truncated unrolling and large batches and achieve similar performance of the CTC model is very promising because all the new advances of the accelerated machines can be applied to further expedite training. For example the NVIDIA cuDNN LSTM is already two times faster than the `tf.LSTMCell` used in this experiments. From the hardware perspectives, our benchmarks show that the latest GPUs like the P100 GPUs are about two times faster than the K40 GPUs to train the LSTM topology used in this paper. The hardware and software advances further reduce the total training time of the sampled CTC to about less than three days.

WER [%]	epoch 10	epoch 20	epoch 40	epoch 160
voice-search	11.4	10.9	10.4	9.6
dictation	9.9	9.0	8.4	7.5
training [days]	1.5	3.6	7.2	14

Table 4. Sampled CTC with Path Counting (truncated unrolling).

Posterior Spikiness: The CTC posteriors are observed to be very spiky since the model prefers output blank symbol most of the time. To measure spikiness of the CTC models, the usual practise is plotting the posteriors for a holdout utterance as in [6]. To compare the spikiness of the sampled CTC and CTC, we computed the average of blank posteriors over whole training corpus. The average of blank posteriors for CTC model is 89%, while this value is 76% and 68% for the sampled CTC with Path Counting and Coin Flipping, respectively. This comparison suggests that the sampled CTC models are not as as sharp as the CTC model.

4. CONCLUSIONS

This paper presented and evaluated sampled CTC as an upper bound approximation of the CTC objective. The sampled CTC connects the CTC constraint to the CE objective through sampling. Two sampling methods were described for which both methods assigns uniform distribution over all possible sample alignments. Theses models are compared mathematically and empirically. The main advantage of the proposed method over CTC was argued to be the computation scalability. Unlike CTC, sampled CTC is a frame-level criterion which allows scalability of training with fixed unrolling and large batches. The sampled CTC was compared with CTC in different setups and it was shown that sampled CTC can achieve the CTC performance quality in significantly shorter training time.

Acknowledgments

The authors would like to thank all members of the Google speech team particularly Matt Shannon and Khe Chai Sim for very useful discussions.

5. REFERENCES

[1] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[2] Lawrence R Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning*, 2016, pp. 173–182.

[4] Kyuyeon Hwang and Wonyong Sung, "Sequence to sequence training of ctc-rnns with partial windowing," in *International Conference on Machine Learning*, 2016, pp. 2178–2187.

[5] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] Haşim Sak, Andrew Senior, Kanishka Rao, Ozan Irsoy, Alex Graves, Françoise Beaufays, and Johan Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4280–4284.

[7] Haşim Sak, Félix de Chaumont Quitry, Tara Sainath, Kanishka Rao, et al., "Acoustic modelling with cd-ctc-smbr lstm rnns," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015, pp. 604–609.

[8] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *arXiv preprint arXiv:1507.06947*, 2015.

[9] Kanishka Rao, Andrew Senior, and Haşim Sak, "Flat start training of cd-ctc-smbr lstm rnn acoustic models," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5405–5409.

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

[11] Ehsan Variani, Tom Bagby, Erik McDermott, and Michiel Bacchiani, "End-to-end training of acoustic models for large vocabulary continuous speech recognition with tensorflow," *Proc. Interspeech 2017*, pp. 1641–1645, 2017.

[12] Albert Zeyer, Eugen Beck, Ralf Schlüter, and Hermann Ney, "Ctc in the context of generalized full-sum hmm training," *Proc. Interspeech 2017*, pp. 944–948, 2017.

[13] Khe Chai Sim and Arun Narayanan, "An efficient phone n-gram forward-backward computation using dense matrix multiplication," *Proc. Interspeech 2017*, pp. 1646–1650, 2017.

[14] Johan Ludwig William Valdemar Jensen, "Sur les fonctions convexes et les inégalités entre les valeurs moyennes," *Acta mathematica*, vol. 30, no. 1, pp. 175–193, 1906.

[15] Golan Pundak and Tara N Sainath, "Lower frame rate neural network acoustic models," *Interspeech 2016*, pp. 22–26, 2016.

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.