IMPROVING CTC USING STIMULATED LEARNING FOR SEQUENCE MODELING

Jahn Heymann[*]

Paderborn University, Germany
heymann@nt.upb.de

Khe Chai Sim, Bo Li

Google AI., USA
{khechai, boboli}@google.com

ABSTRACT

Connectionist temporal classification (CTC) is a sequence-level loss that has been successfully applied to train recurrent neural network (RNN) models for automatic speech recognition. However, one major weakness of CTC is the conditional independence assumption that makes it difficult for the model to learn label dependencies. In this paper, we propose stimulated CTC, which uses stimulated learning to help CTC models learn label dependencies implicitly by using an auxiliary RNN to generate the appropriate stimuli. This stimuli comes in the form of an additional stimulation loss term which encourages the model to learn said label dependencies. The auxiliary network is only used during training and the inference model has the same structure as a standard CTC model. The proposed stimulated CTC model achieves about 35 % relative character error rate improvements on a synthetic gesture keyboard recognition task and over 30 % relative word error rate improvements on the Librispeech automatic speech recognition tasks over a baseline model trained with CTC only.

***Index Terms*—** connectionist temporal classification, stimulated learning, sequence classification

1. INTRODUCTION

Natural languages exhibit a hierarchical structure where (discrete) abstractions are composed of lower level entities which can be either discrete or continuous. For example, words are used to convey a semantic meaning and are composed of characters or symbols. These symbols can be communicated in written or oral form, where the representation is again changed and we observe a two-dimensional picture or an audio signal, i.e. a continuous representation.

Despite this hierarchical structure, nowadays recognition models usually model each level separately and then fuse the respective probabilities. For example, the acoustic model is usually not explicitly aware of the concept of words. Hybrid models assume a conditional independence given the current state and connectionist temporal classification (CTC) [1] assumes independence of each symbol given the whole audio signal. Although recurrent neural networks could theoretically resolve the label dependencies and thus implicitly learn linguistic concepts such as words, practical issues with learning long-term dependencies and limited model capacity arguably prevent them from doing so [2].

In this work, we aim to make the model aware of the context of the higher level representation by explicitly incorporating it during training. Assuming we know the alignment between the different representations, we utilize the so called *stimulated learning* [3, 4, 5] to constrain the state trajectory of a recurrent neural network (RNN) model to conform that of a higher level model, i.e. we stimulate the states at segment boundaries to be the same as those provided by an additional recurrent model of the higher level representation. In particular, an acoustic model is stimulated to conform the state trajectory of a language model (LM) by gradually transitioning from one LM state to another one while consuming the acoustic frames as observations. This also has the benefit that we know the state of the model at certain points in time and can train it in a segment-wise fashion – similar to truncated BPTT but without any loss of gradient information. This stimulation happens only during training time and no additional overhead during inference is introduced.

For many applications, however, we do not have such knowledge about (meaningful) segment boundaries but rather need to infer those from the observation itself. For these cases, we extend our model to full sequence training. Namely, we propose to combine it with CTC. To this extend we utilize the CTC state posteriors, i.e. the soft alignment, to obtain an attention over the state trajectory to stimulate at the right positions in time. We hypothesize that this helps the model to exploit linguistic structure and to attenuate the conditional independence assumption introduced by CTC by stimulating the model to implicitly learn label dependencies.

Our model bears many similarities to the hierarchical recurrent neural network (HRNN) [6] and even more to the hierarchical multiscale recurrent neural network (HM-RNN) [7, 8] which also tries to exploit the hierarchical structure of the data. The latter extended RNN consists of multiple layers and a latent variable which controls the current operation for each layer. Higher layers *copy* their state until the layer below *flushes* its content. In this case, the upper layer updates its state and provides the updated state as a context for the next step to the layer below. Compared to the segment-wise model we propose, the functionality and rational is very similar if the HM-RNN would learn to flush at the same segment boundaries. In fact, [8] also uses phoneme boundaries as information to guide the latent variable during training with an additional loss term. For the full sequence training, our proposed mechanism to discover boundaries is different and more explicit as it exploits prior knowledge (for example about words in the transcription).

The remainder of this paper is organized as follows. Section 2 introduces the formulation of stimulated CTC. Section 3 describes the training procedures of the proposed models. Finally, section 4 presents experimental results on a gesture keyboard recognition task and the Librispeech [9] automatic speech recognition tasks.

2. STIMULATED CTC

In a label sequence prediction task, the input feature sequence ($\boldsymbol{X}$) and the output label sequence ($\boldsymbol{W}$) are not always of the same length. A prediction model typically introduces a time-aligned label sequence containing repeats and blank symbols, $\boldsymbol{Y}$, so that the probability of the label sequence given the input sequence, $P(\boldsymbol{W}|\boldsymbol{X})$,

[*]Work performed while at Google.

can be decomposed into a product of conditional probabilities. For a unidirectional model, we have

$$P(\boldsymbol{W}|\boldsymbol{X}) = \prod_{t=1}^{T} P(y_t|\boldsymbol{X}_{1:t}, \boldsymbol{W}_{1:k_t}) \tag{1}$$

where y_t is the time-aligned label at time t, $\boldsymbol{X}_{1:t}$ is the input feature sequence up to time t and $\boldsymbol{W}_{1:k_t}$ is the corresponding output label sequence up to label k_t (the last label whose end time is less than t).

CTC is a popular sequence prediction model that has been successfully applied to automatic speech recognition [1] and keyboard gesture recognition [10]. CTC assumes that the conditional probabilities are independent of label history:

$$P(y_t|\boldsymbol{X}_{1:t}, \boldsymbol{W}_{1:k_t}) \approx P(y_t|\boldsymbol{X}_{1:t}) = P(y_t|\boldsymbol{h}_t^{(x)}) \tag{2}$$

where $\boldsymbol{h}_t^{(x)}$ is the encoded input features at time t (typically using an RNN) to capture long term input history. Therefore, it has a limited capacity to learn a language model. In practice, context-dependent output units are used in combination with an external LM [11]. There are other end-to-end techniques that model the label dependencies explicitly, such as recurrent neural network transducer (RNN-T) [12, 13], listen attend spell (LAS) [14, 15] and neural transducer (NT) [16]. Specifically, RNN-T can be viewed as an extension to CTC by explicitly incorporating label dependencies:

$$P(y_t|\boldsymbol{X}_{1:t}, \boldsymbol{W}_{1:k_t}) \approx P(y_t|\boldsymbol{h}_t^{(x)}, \boldsymbol{h}_{k_t}^{(w)}) \tag{3}$$

where $\boldsymbol{h}_{k_t}^{(w)}$ is the RNN encoded LM state, summarizing the label history up to label k_t. However, label history has to be explicitly tracked and there is no merging of LM states due to the continuous state representation of an RNN.

In this paper, we investigate the possibility of incorporating the label history information ($\boldsymbol{h}_{k_t}^{(w)}$) implicitly into $\boldsymbol{h}_t^{(x)}$ through stimulated learning [3, 4, 5] to improve a CTC model. The *stimuli* are generated by an auxiliary RNN, which is jointly learned with the prediction RNN. We treat $\boldsymbol{h}_{k_t}^{(w)}$ as *privileged information* that are only available during training and introduce an additional *stimulation loss* term to minimize the mean squared error (MSE) between $\boldsymbol{h}_t^{(x)}$ and $\boldsymbol{h}_{k_t}^{(w)}$. Stimulated CTC can be viewed as a kind of distillation (student-teacher) training [17], where the auxiliary RNN LM is used as a teacher to guide the student CTC RNN to learn a better underlying state representation. Like RNN-T, stimulated CTC jointly trains the RNN LM with the recognition RNN model but the LM component is only used during training.

A schematic overview of the stimulated CTC model is shown in Fig. 2.

3. TRAINING PROCEDURE

The stimulated CTC loss function consists of three loss terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{ctc}} + \alpha\mathcal{L}_{\text{lm}} + \beta\mathcal{L}_{\text{stimu}}$$

where $\mathcal{L}_{\text{ctc}}$ is the standard CTC loss, $\mathcal{L}_{\text{lm}}$ is the LM loss and $\mathcal{L}_{\text{stimu}}$ is the stimulation loss. α and β are weight factors that can be adjusted to trade-off the importance of each loss. The LM loss, which is the standard cross-entropy loss for RNN LM [18], is given by:

$$\mathcal{L}_{\text{lm}} = -\frac{1}{K}\sum_{k=1}^{K} \log P(w_k|\boldsymbol{h}_{k-1}^{(w)})$$

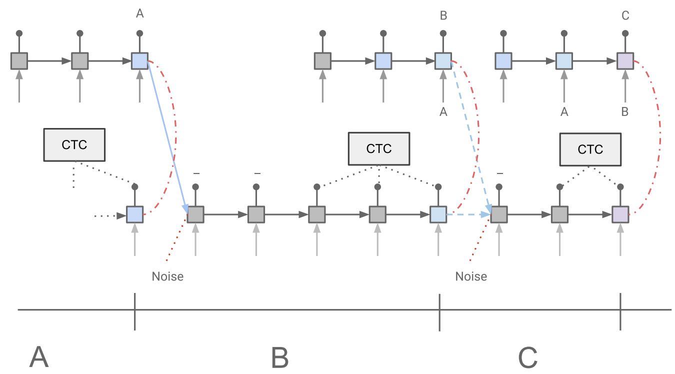

Fig. 1. Illustration of the segment-level model with the recognition (lower part) and the auxiliary (n-gram) LM (upper part). The dashed blue lines at the B-C boundary indicate that the state can either come from the auxiliary model or from the observation model itself (for m segment-level training). The red dash-dot lines show where the MSE loss is applied. Noise is added optionally to the context as indicated by the dashed red line.

As previously mentioned, $\boldsymbol{h}_{k-1}^{(w)}$ is encoded using an auxiliary RNN model. The RNN can also be constrained to resemble an n-gram LM [19] by limiting the RNN memory to the last $n-1$ labels. This allows us to control the complexity of the auxiliary model. The LM loss ensures that $\boldsymbol{h}_{k-1}^{(w)}$ is highly predictive for w_k. Using RNN for the auxiliary model makes it easy to jointly optimize both the recognition and auxiliary models.

The stimulation loss, $\mathcal{L}_{\text{stimu}}$, helps the recognition model learn the higher level (i.e. linguistic) abstractions implicitly by ensuring that the RNN state trajectory of the recognition model conforms to that of the auxiliary RNN. The stimulation loss can be computed in different ways depending on the whether a hard or soft alignment is used. For most sequential observations a meaningful alignment (segmentation) is unavailable at training time and has to be inferred as part of the training process.

3.1. Hard Alignment

When hard alignment is used, we can apply the stimulation loss at the segment boundaries:

$$\mathcal{L}_{\text{stimu}} = \frac{1}{K}\sum_{k=1}^{K} ||\boldsymbol{h}_{\tau_k}^{(x)} - \boldsymbol{h}_k^{(w)}||^2 \tag{4}$$

where K is the label sequence length and τ_k is the end of segment boundary for the k-th label. This allows us to perform segment-level training by splitting the observation sequence ($\boldsymbol{X}_{1:T}$) into K non-overlapping segments, S_k, one for each label. For the k-th segment, we train an RNN[1] to predict w_k at the end of the segment and *blanks* elsewhere. $\boldsymbol{h}_{k-1}^{(w)}$ is used as the initial RNN state and the final RNN state after observing the segment S_k is constrained to be close to $\boldsymbol{h}_k^{(w)}$ (by using the stimulation loss in Eq. 4). If the constraint is well-satisfied, inference becomes easy as we can just treat the model as a normal RNN and fully unroll the whole sequence.

[1]We use the term RNN here as a general description of a recurrent model. Any specific architecture like LSTMs, GRUs etc. can be used.

The above segment-level training decomposes a full sequence optimization problem (due to the recurrent nature of RNNs) into smaller independent optimization problems. This is similar in spirit to the work that decomposes optimization of a deep neural network into independent optimization problems, one for each layer [20, 21].

In the following, we will discuss several improvements made to the above segment-level training.

- **Stochastic Stimulation**
  It is unlikely that the recognition model will be able to transition to these states *exactly* since the observations themselves are assumed to be continuous. To make the model more robust against small errors in the LM state estimation, we add small Gaussian noise to the initial states during training. The variance can either be fixed upfront or generated by the auxiliary model. In the latter case, instead of MSE, negative log likelihood is used as the stimulation loss. Therefore, if the prediction of the model is inaccurate, the auxiliary model is encouraged to increase the variance. Ultimately, this should result in a variance which reflects the model uncertainty about its decision but keeps the context informative.
- **Multi-label Segments**
  Stochastic stimulation alone does not address the problem with inaccurate state trajectory estimation. When trained in a segment-wise fashion, the model never encounters a true cross segment transition since the initial state for each segment is always provided by the auxiliary model. This leads to a significant decrease in performance as we will show in the results section. To circumvent the problem, we consider training segments of multiple labels by stitching together m consecutive segments. This way, we expose the model to situations as described above during training and the model learns to cope with uncertainty in the context and also to recover from classification errors.
- **Constrained CTC loss**
  Another problem we encountered during initial experiments is that the model struggles to output the prediction at the very end of the segment. Using a per-segment CTC loss led to another problem where the model quickly learned to make the prediction at the first frame of the segment based on the LM state provided by the auxiliary model, without considering the observation. In order to prevent this behaviour, we use a constrained CTC loss that only allows labels to be emitted in the last $25\,\%$ frames of the segment, forcing the model to output blank symbols at the beginning.

3.2. Soft Alignment

Soft alignments are computed when computing the gradient of the CTC loss with respect to the logits. This is given by, $\gamma_t(k) = P(y_t = w_k|\boldsymbol{X}_{1:T})$, the probability that a label w_k is aligned to time t, which can be computed efficiently using the forward-backward algorithm [22]. We use this information as weight to calculate the stimulation loss as follows:

$$\mathcal{L}_{\text{stimu}} = \frac{1}{K \cdot T} \sum_k \sum_t \gamma_t(k) ||\boldsymbol{h}^{(x)}{}_t - \boldsymbol{h}^{(w)}{}_k||^2$$

Note that for soft-alignment stimulation, we unroll the recognition network over the whole sequence. This also means, that we have only one CTC loss over the whole sequence and not one per segment as in the previously described segment-level training. Overall, this is much closer to a standard RNN model, except for the stimulation of the state, which is meant to guide its trajectory.

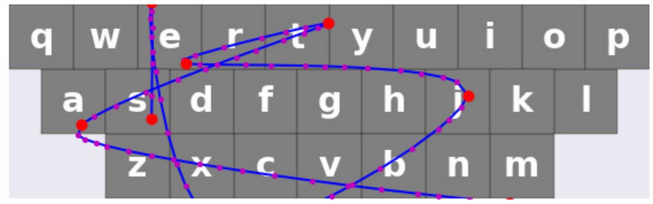

Fig. 2. Example input for keyboard gesture recognition. The red dots illustrate the x-y-coordinates whicht are used as inputs. The big red dots correspond to segment boundaries.

4. EXPERIMENTS

To analyze the behavior and evaluate the performance of our proposed approach we conduct experiments on two different tasks. All the models are trained using Tensorflow [23] using the efficient CTC implementation described in [22].

The first one is keyboard gesture recognition. For this task, the input are the x-y-coordinates of a swipe gesture performed on a virtual keyboard and the goal is to predict the *swiped* word. We choose this task because it allows us to easily generate data with known segmentation so that we can perform in-depth evaluation to compare the segment-level (Subsec. 3.1) and the full sequence training methods (Subsec. 3.2).

The second task is automatic speech recognition (ASR) for which we use the Librispeech database [9] as an example. Here, we cannot use the segment-level model and focus on the full sequence model which we compare to a model without stimulation.

4.1. Keyboard gestures recognition

For the keyboard gesture recognition scenario, the goal is to predict a word given the x-y-coordinates from a swipe gesture on a virtual keyboard. All data for this task is generated by sampling a word (with at least two characters) from the CMU dictionary, and connecting the keys of a virtual keyboard corresponding to the words characters.

To connect the keys we use Bézier curves and add three types of noise. One, which we call anchor noise, moves the connection point away from the center of the key. We use a Gaussian distribution with zero mean and a fixed variance to model this type of noise. The second one influences the sampling interval of the curves and is again modeled using a gaussian distribution. The last one modifies the distance of the control points for the construction of the Bézier curves and thus influences the curvature.

An example from this dataset is depicted in Fig. 4.1. We split the dataset into one for training, validation and evaluation by using $80\,\%$, $10\,\%$ and $10\,\%$ of the words from the CMU dictionary respectively, resulting in 88713, 11089 and 11089 unique words for the sets. We further sample all three noise types during training for each iteration randomly and with a fixed seed for the validation and evaluation set.

For all experiments we use a one layer LSTM network with 256 units as the recognition network and auxiliary model. Our baseline is a LSTM network of the same size trained with CTC without stimulation. Note that this model is exactly equivalent in terms of inference with the stimulated models. We train all models for 4M iterations and use the one with the best performance on the validation set for evaluation. All networks are trained to predict the strokes of the gesture, i.e. the start and end key. The validation set is also

Table 1. CERs / % on the noisy gesture recognition dataset for different number of combined segments.

	Combined segments			
	1	2	4	6
Stimulated + fixed noise	38.8	6.2	4.7	4.3
Stimulated + learned noise	31.3	6.0	4.2	3.6

Table 2. CERs / % on the noisy gesture recognition dataset with full sequence training. *Stimulated* uses a priori boundary information for stimulation while *stimulated CTC* uses the CTC soft-alignment as attention.

CTC	Stimulated	stimulated CTC
5.5	3.6	3.7

used to choose the optimal parameters for the learning rate, the loss weights, the context noise variance and the n-gram order of the auxiliary model if applicable. For evaluation, we decode using a prefix beam search decoder constrained by all words contained in the CMU dictionary. The metric we are interested in is the character error rate (CER). To minimize the effect of randomness, we run each experiment at least three times. We always report the best result here but none of the results had notable outliers.

The baseline model achieves a CER of $5.5\,\%$ for this task (see Tbl. 2). The basic model (with fixed context noise) performs very poorly, resulting in a CER of $38.8\,\%$. However, if we provide the correct context vector also during evaluation, the CER reduces drastically to $1.9\,\%$. This supports what we suggested in Subsec. 3.1: Once the prediction is off for one segment, the model might not be able to recover due to the way we use it during inference. The proposed solution for this was to combine a certain number of segments to also expose the model to such situation already during training time. And indeed, if we only combine every two segments, the CER goes down to $6.2\,\%$. We also observe that the performance improves the more segments we combine. Notably, the model outperforms the baseline when combining four segments or more. If we also let the auxiliary model learn the parameters of the additive context noise and combine six segments, the model reaches a CER of $3.6\,\%$, i.e. an improvement of nearly $35\,\%$ relative compared to the baseline. All results are shown in Tbl. 1.

Finally, we evaluate the full sequence model. The previous results already showed, that the performance improves with the number of combined segments. Tbl. 2 shows the results for fully unrolled training, i.e. when we combine all segments of a gesture. Here, no context is provided by the auxiliary model and it is only used to stimulate the state at boundaries. For a better comparison with the baseline, we also do not add noise to the context in this case. As the results show, this does not influence the performance of the stimulated model and the CER is still $3.6\,\%$. We now omit the a priori knowledge of the boundaries and use the CTC soft-alignment to stimulate the model (Subsec. 3.2) The performance is only marginally effected ($3.7\,\%$ CER) by this, showing the effectiveness of our proposed approach.

Table 3. WERs / % on the Librispeech datasets using different decoding strategies.

Decoding Strategy	Model	dev		test	
		clean	other	clean	other
Greedy	CTC	19.1	35.2	20.0	36.0
	CTC stimulated	12.3	27.1	12.7	27.7
0-gram	CTC	17.6	34.1	18.5	34.8
	CTC stimulated	11.3	26.2	11.7	26.9
3-gram	CTC	14.6	29.7	15.2	30.4
	CTC stimulated	9.8	23.6	10.3	24.2

4.2. ASR: Librispeech

To further evaluate the full sequence model, we also compare it to a CTC baseline on the Librispeech ASR task [9]. We use all available data for training (i.e. all three training sets for a total of $960\,\mathrm{h}$ of speech) and evaluate on the *clean* and *other* set.

The baseline model has xx LSTM layers with xx units each and is trained using the CTC loss only. Instead of characters or phonemes, we choose subword units [24] as targets, leaving us with a total of xx classes. We compare this model to a stimulated one where we use the exact same structure but stimulate the last layer as described in Subsec. 3.2. Consequently, the auxiliary is a one layer LSTM with the same number of units and acts as an LM for the subword units. Both models are trained with Adam optimization [25] with a learning rate of $1\mathrm{e}^{-5}$.

The results for both model decoded with different language constraints are shown in Tbl. 3. For 0-gram decoding, a zero LM weight is used to constraint the decoder to output valid words. For all cases, the stimulated model achieves much better results with gains around $30\,\%$. Both models perform at least twice as good on the *clean* set compared to the *other* set while the stimulated model loses a bit more than the non-stimulated one. Comparing the different decoding constraints, the stimulated model also does not profit as much from a stronger linguistic constrain as does the vanilla CTC model. This can be seen as an indication that the model is able to better exploit linguistic structure due to the stimulation as hypothesized.

Although our preliminary experimental results show promising improvements, we acknowledge that our results are generally worse compared to other work (e.g. [9]) and the baseline model is weak. We suspect that this is due to untuned hyperparameters in combination with the choice of subword as the output units. In addition, it may also be difficult to learn a CTC model from scratch with long utterances. Further experiments are needed to validate the effectiveness of the proposed method.

5. CONCLUSIONS

In this paper, we proposed using stimulated learning to improve a CTC model by introducing additional loss terms to encourage the model to learn implicit label dependencies. This can be viewed as a special form of student-teacher training where an RNN LM is used as a teacher to help the student CTC model learn the underlying RNN states at label emission points. Preliminary experimental results on a synthetic gesture keyboard input recognition task and the Librispeech automatic speech recognition tasks show that the proposed stimulated learning has promising potential in learning a better CTC model.

6. REFERENCES

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning (ICML)*, 2006.

[2] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, 2013.

[3] S. Tan, K. C. Sim, and M. Gales, "Improving the interpretability of deep neural networks with stimulated learning," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015.

[4] C. Wu, P. Karanasou, M. JF Gales, and K. C. Sim, "Stimulated deep neural network for speech recognition," in *Proc. Interspeech*, 2016.

[5] C. Wu, M. J. F. Gales, A. Ragni, P. Karanasou, and K. C. Sim, "Improving interpretability and regularization in deep learning," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 26, no. 2, 2018.

[6] K. Hwang and W. Sung, "Character-level language modeling with hierarchical recurrent neural networks," in *v*, 2017.

[7] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[8] J. Park, I. Choi, Y. Boo, and W. Sung, "Hierarchical recurrent neural networks for acoustic modeling," in *Proc. Interspeech*, 2018.

[9] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[10] O. Alsharif, T. Ouyang, F. Beaufays, S. Zhai, T. Breuel, and J. Schalkwyk, "Long short term memory neural network for keyboard gesture decoding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015.

[11] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *arXiv preprint arXiv:1507.06947*, 2015.

[12] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[13] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017.

[14] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[15] C.-C. Chiu, T. N Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J Weiss, K. Rao, K. Gonina, et al., "State-of-the-art speech recognition with sequence-to-sequence models," *arXiv preprint arXiv:1712.01769*, 2017.

[16] E. Battenberg, J. Chen, R. Child, A. Coates, Y. Gaur, Y. Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, "Exploring neural transducers for end-to-end speech recognition," *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017.

[17] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[18] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[19] C. Chelba, M. Norouzi, and S. Bengio, "N-gram language modeling using recurrent neural network estimation," *arXiv preprint arXiv:1703.10724*, 2017.

[20] M. A Carreira-Perpinán and M. Alizadeh, "Parmac: distributed optimisation of nested functions, with application to learning binary autoencoders," *arXiv preprint arXiv:1605.09114*, 2016.

[21] R. Raziperchikolaei and M. A Carreira-Perpinán, "Optimizing affinity-based binary hashing using auxiliary coordinates," in *Advances in Neural Information Processing Systems*, 2016.

[22] K. C. Sim, A. Narayanan, T. Bagby, T. N Sainath, and M. Bacchiani, "Improving the efficiency of forward-backward algorithm using batched computation in tensorflow," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S Corrado, A. Davis, J. Dean, M. Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[24] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[25] D. P Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.