INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Towards using context-dependent symbols in CTC without state-tying decision trees

Jan Chorowski, Adrian Lancucki, Bartosz Kostka, Michal Zapotoczny

University of Wrocław, Poland
jch@cs.uni.wroc.pl

Abstract

Deep neural acoustic models benefit from context-dependent (CD) modeling of output symbols. We consider direct training of CTC networks with CD outputs, and identify two issues. The first one is frame-level normalization of probabilities in CTC, which induces strong language modeling behavior that leads to overfitting and interference with external language models. The second one is poor generalization in the presence of numerous lexical units like triphones or tri-chars. We mitigate the former with utterance-level normalization of probabilities. The latter typically requires reducing the CD symbol inventory with state-tying decision trees, which have to be transferred from classical GMM-HMM systems. We replace the trees with a CD symbol embedding network, which saves parameters and ensures generalization to unseen and undersampled CD symbols. The embedding network is trained together with the rest of the acoustic model and removes one of the last cases in which neural systems have to be bootstrapped from GMM-HMM ones.

Index Terms: LSTM, CTC, context dependent phones, state tying, decision trees

1. Introduction

Acoustic models built with end-to-end trained deep neural networks are general enough to read raw waveforms [1] and output characters [2], bypassing steps of feature extraction and preparation of pronunciation lexicons. However, even with abundant training data, the performance of end-to-end networks can be improved by introducing elements from classical GMM-HMM models, most notably the concept of context-dependent (CD) output symbols. To limit the number of considered states, these are clustered using state-tying decision trees [3, 4], often taken from GMM-HMM models, making for better representations of CD symbols which are rare or absent in the training data.

In this work we consider Connectionist Temporal Classification (CTC) [5] networks with context dependency: for each frame of acoustic features, the network predicts CD targets (phonemes or characters in context). To the advantage of CTC networks, the training procedure can be started on unaligned transcripts and does not require bootstrapping from a GMM-HMM system. To preserve these properties, we propose to compute the representations of CD output symbols with an embedding network, which is a neural analogue to a state-tying decision tree. The embedding network naturally exploits similarities between contexts, and generalizes to previously unseen ones.

We address the challenges posed by using CD symbols with CTC. We show that using CD symbols improves over a vanilla CTC solution, and that the best results are obtained with the formulation of CTC loss more similar to the Lattice-Free MMI methods. During training, it normalizes probabilities on the transcript level rather than on the frame level.

2. Background

2.1. GMM-HMMs with Decision Trees

The classical GMM-HMM acoustic model assumes that each speech segment (typically a 25ms long acoustic frame) is emitted from a single Gaussian mixture specified by the hidden state of the HMM [6, 7]. This simple emission model assumes similarity of frames emitted from the same state, and dissimilarity to the remaining frames. It is achieved by careful modeling of acoustic phenomena. Each phoneme is partitioned into three sub-phonemic states, and context-dependent changes in pronunciation (e.g., voicing) are handled by differentiating between the emissions of the same phoneme from different contexts.

Particularly popular are triphones, i.e., phonemes considered in their left and right contexts. Larger units like quinphones are possible as well. To cope with the large amount of possible CD symbols the emission GMMs are shared between contexts, with the mapping performed by state-tying decision trees [8]. Typically, such trees map each CD symbol to one of a few thousand GMMs (also called tied states).

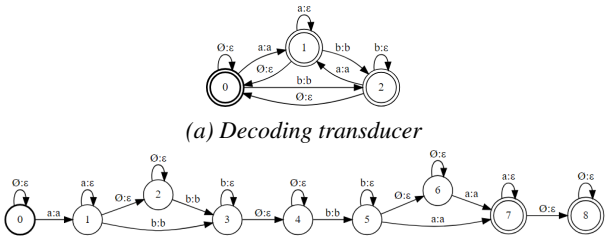
In a hybrid DNN-HMM model the Gaussian mixture representation of speech frames is replaced with a deep neural network that is tasked with predicting the HMM state aligned with each frame. In principle, the network can become invariant to many acoustic phenomena and it is no longer necessary to model them explicitly. Indeed, little to no gains are reported for dividing phonemes into subphonemic states [3]. However, explicit modeling of the context still improves recognition accuracy [3, 4]. For this reason, a typical DNN-HMM system employs neural networks that predict the tied states of a previously trained GMM-HMM model. Consequently, the application of neural networks still depends on decision trees that map CD symbols to individual network outputs.

2.2. End-to-end approaches to speech recognition

End-to-end systems can be trained from scratch without a dependency on a previously built GMM-HMM system. They employ cost functions able to establish an alignment between the sequence of acoustic frame features and the elements of the target transcript. This can be accomplished in two ways. First, a sequence-level cost function can be applied to an HMM topology. Examples of such models are CTC [5, 9], Lattice-Free Maximum Mutual Information (LF-MMI) [10, 4] and Graph Transformer Networks (GTN) [11, 12]. A second family of models replaces HMMs with a neural attention mechanism [13, 14]. In this contribution we focus on CTC due to its popularity and wide adoption in recent ASR projects such as Baidu's DeepSpeech [2] or the Mozilla Speech to text engine[1]. To better understand the interaction of the CTC loss with CD symbols, we will describe CTC from the graph transformer point of view.

[1] https://github.com/mozilla/DeepSpeech

http://dx.doi.org/10.21437/Interspeech.2019-2720



(a) Decoding transducer

(b) Training graph for a single utterance "abba"

Figure 1: *CTC transducers for a two-letter alphabet* $\mathcal{L} = \{a, b\}$

2.3. Locally and globally normalized CTC

Let Y be a desired transcript, i.e., a sequence of symbols over an alphabet $\mathcal{L}$. An extended transcript $\hat{Y}$ of symbols from the alphabet $\hat{\mathcal{L}} = \mathcal{L} \cup \{\emptyset\}$ is a longer sequence formed by repeating elements of Y and padding them with a special blank ($\emptyset$) token. Consecutive elements of $\hat{Y}$ correspond to input frames. Y can be uniquely recovered from $\hat{Y}$ by removing repetitions and blanks, implemented by a function $B(\hat{Y}) = Y$ [5]. This operation can also be implemented using a Finite State Transducer (FST) [15] shown in Figure 1. Equivalently, all extended transcripts that map to Y_e can be captured with a regular expression

$$\emptyset^*\ a\ a^*\ \emptyset^*\ b\ b^*\ \emptyset\ \emptyset^*\ b\ b^*\ \emptyset^*\ a\ a^*\ \emptyset^*\ ,$$

where the * operator denotes zero or more repetitions.

The network reads a sequence of T acoustic frames and computes a matrix $O \in \mathbb{R}^{T \times |\hat{\mathcal{L}}|}$ of unnormalized, non-negative scores $O_{t,l}$ for emitting symbol l from frame t. The unnormalized score of an entire extended transcript $\hat{Y}$ is defined as the product of scores assigned to all its symbols

$$\hat{S}(\hat{Y}) = \prod_t O_{t,\hat{Y}_t}, \tag{1}$$

often replaced by a sum of logarithms for numerical stability.

2.3.1. CTC-G: Globally normalized CTC

To compute the probability assigned to a transcript Y we normalize the sum of scores of all extended transcripts that map to Y by the sum of scores of all extended transcripts, as advocated in GTN [11], LF-MMI [10], and Alphanet [16]:

$$P(Y|X) = \frac{\sum_{\hat{Y} \in B^{-1}(Y)} \hat{S}(\hat{Y})}{\sum_{\hat{Y} \in \hat{\mathcal{L}}^T : \hat{Y} \text{ is valid}} \hat{S}(\hat{Y})}. \tag{2}$$

Both the numerator and the denominator can be computed using the forward algorithm over graphs unrolled over utterance frames: the utterance graph (Figure 1b) for the numerator and the decoding graph (Figure 1a) for the denominator. The time complexity depends on the number of states in the graph. For CD systems it is dominated by the denominator computation.

2.3.2. CTC: Local normalization in CTC

CTC avoids denominator computation in (2) by using framewise normalization of scores. Observe that for any Z_t we have

$$P(Y|X) = \frac{\sum_{\hat{Y} \in B^{-1}(Y)} \hat{S}(\hat{Y})}{\sum_{\hat{Y} \in \hat{\mathcal{L}}^T : \hat{Y} \text{ is valid}} \hat{S}(\hat{Y})} = \frac{\sum_{\hat{Y} \in B^{-1}(Y)} \prod_t O_{t,\hat{Y}_t}/Z_t}{\sum_{\hat{Y} \in \hat{\mathcal{L}}^T : \hat{Y} \text{ is valid}} \prod_t O_{t,\hat{Y}_t}/Z_t}.$$

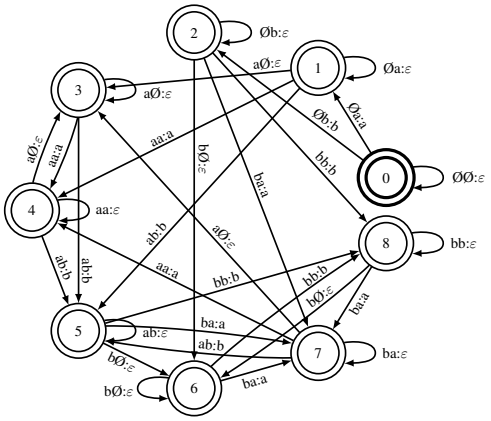


Figure 2: *Bi-character CTC decoding transducer for a two-letter alphabet* $\mathcal{L} = \{a, b\}$. *It imposes overlap between subsequent bi-characters.*

Validity of an extended transcript refers to the correct overlap of neighboring symbols, which is an issue if CD symbols are used. With ordinary context independent (CI) symbols, every extended transcript is valid. Taking $Z_t = \sum_y O_{t,y}$ we can locally normalize network outputs into probabilities $p(\hat{Y}_t = y) = O_{t,y}/Z_t$. Furthermore, when all extended transcripts are valid, which is the case when context-independent (CI) symbols are used, the denominator $\sum_{\hat{Y} \in \hat{\mathcal{L}}^T} \prod_t p(\hat{Y}_t = y)$ is always 1.0 and need not be computed recovering the familiar CTC formula

$$P(Y|X) = \sum_{\hat{Y} \in B^{-1}(Y)} \prod_t p(\hat{Y}_t|X). \tag{3}$$

2.4. CTC loss with context-dependent symbols

We illustrate the CTC criterion with CD symbols on an example. Let $[a, b, b, a]$ be the target transcript, which corresponds to the sequence $Y_E = [\emptyset a, ab, bb, ba]$ of bi-characters with $\emptyset a$, aa, and ba denoting CD-variants of a. The extended alphabet has 7 symbols (1 blank and 6 CD symbols). All extended transcripts of Y_E match the expression

$$\emptyset^*\ \emptyset a\ \emptyset a^*\ \emptyset^*\ ab\ ab^*\ \emptyset^*\ bb\ bb^*\ \emptyset^*\ ba\ ba^*\ \emptyset^*.$$

Please note that the blank symbol between repeated bs became optional, because the two emissions of b have different contexts.

From the example we can see that CD symbols which form an extended transcript overlap with the symbol at frame t, setting the context for frame $t+1$. This means that strings over the alphabet $\hat{Y}$ with improper overlaps are invalid and must be removed from the denominator sum in (2). This can be achieved with the forward algorithm applied to a sparsely connected decoding graph, e.g., the bi-character one in Figure 2.

However, the locally normalized CTC loss (3) implicitly sums over all strings in $\hat{Y}^*$, including the invalid ones whose CD symbols do not overlap. This makes the network prone to overfitting, because it must learn to properly overlap the symbols that are predicted and thus remove all ambiguity from its outputs. In practice this means that CTC predictions will become very sharp, selecting for each frame only one CD-symbol with high probability. The model is forced to start modeling the language in order to be able to essentially output a single hypothesis. During decoding, this internal language model conflicts the external one, requiring the use of low acoustic model weights (cf. Figure 4).

Under a different interpretation, local normalization forces the network to differentiate between the same symbol in different contexts, even though they may correspond to exactly the same sound. Clustering CD-symbols using decision trees [3] helps by grouping similarly sounding tied symbols. However, even with tied symbols, there are many invalid extended transcripts which the network will learn not to emit.

2.4.1. CTC-GB: context-dependent blanks in CTC-G

Globally normalized CTC allows to introduce multiple context-dependent blank characters (**CTC-GB**). This brings the CTC topology closer to a classical tri-state HMM, where blanks serve as secondary subcharacter states. In our running example Y_E, all valid extended transcripts with CD blanks fit the pattern

$$\emptyset\emptyset^* \ \emptyset a \ \emptyset a^* \ a\emptyset^* \ ab \ ab^* \ b\emptyset^* \ bb \ bb^* \ b\emptyset^* \ ba \ ba^* \ a\emptyset^*,$$

where $\emptyset\emptyset$, $a\emptyset$, and $b\emptyset$ are the CD blank symbols. We found that the CD blanks are especially helpful when we tie the prototypes used by the scoring layer in the network, as described next.

3. CD Embeddings for End-to-end Training

The last parametrized layer of a typical deep neural model is a linear layer, also called an embedding or a look-up layer. It stores a prototype vector per each of N output symbols and has $\mathcal{O}(N)$ parameters. Because CD targets are strings of D symbols from the alphabet $\hat{\mathcal{L}}$, the embedding layer naively requires $\mathcal{O}(|\hat{\mathcal{L}}|^D)$ parameters. This exponential growth with D can be alleviated by fixing the number of tied states, and mapping each of $|\hat{\mathcal{L}}^D|$ strings to a tied state in the scoring layer with a decision tree.

We propose an alternative way to handle large numbers of output symbols: keep the full set of output symbols (which still grows exponentially with the size of the context), but make their prototypes co-dependent by generating them with an auxiliary Context-Dependent Embedding (CDE) neural network, analogous to a state-tying decision tree. CDE reduces the number of parameters in the scoring layer, enables learning similarities between contexts, and generalization to previously unseen ones. Prototypes of n-gram CD symbols (in our case bi-chars and tri-chars) are computed with n separate character embedding layers. Those embeddings of individual characters are then concatenated and passed through a ReLU MLP (Figure 3).

4. Model Description

We use the Wall Street Journal dataset, with the typical split into *Si284*, *Dev93*, and *Eval92* as train, dev and test sets, respectively. We calculate 80-dim filterbank features along with the energy in each frame, extend them with temporal Δs and Δ-Δs, and apply global cepstral mean and variance normalization (CMVN). We pre-process the data using Kaldi [17], implement the neural

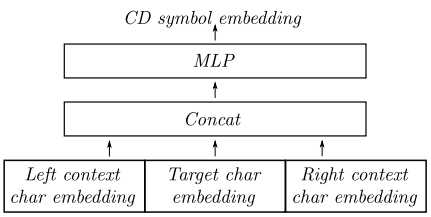


Figure 3: *Architecture of the Context-Dependent Embedding (CDE) network. Embeddings of individual characters are combined to form a CD symbol embedding.*

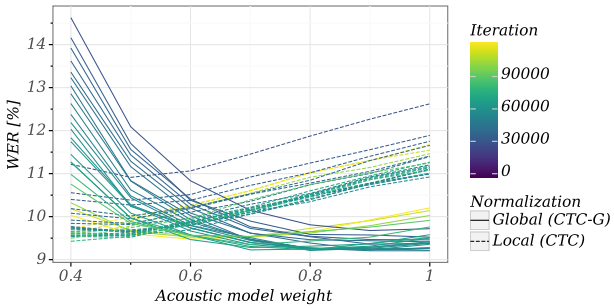


Figure 4: *Globally (CTC-G) versus locally normalized CTC with CD symbols under external LM. CTC-G reaches lower WER with higher acoustic weights than CTC, indicating that its outputs are more ambiguous and conflict less with the LM. The range of optimal acoustic weights is also broader for CTC-G.*

network in PyTorch [18] and use Kaldi's FST decoder.[2]

Our model has two convolutional layers with ReLU activations, dimensionality 32, kernel sizes 7×7, and strides 1×2 and 3×1, meaning that the first layer halves the resolution along the frequency axis and the second layer reduces the length of the utterance. We apply batch normalization [19] after each convolution. The convolutions are followed by four BiLSTM layers with 320 cells each. Therefore, scoring layer prototype vectors are also 320 dimensional. The alphabet has $|\hat{\mathcal{L}}| = 49$ symbols, which accounts for 2401 bi-characters, and over $117k$ possible tri-characters. We limit the set of tri-characters from $117k$ to less than $18k$ which appear in the training data and the language model. We treat the leftmost and rightmost characters of a tri-char as left and right contexts. The context-dependent scoring network embeds each CI symbol into 160 dimensions for bi-chars and 110 dimensions for tri-chars, then uses 2 ReLU layers with 320 units and an affine projection into 320 dimensions.

Hyperparameters were adjusted on the baseline CTC context-independent model. All models are trained with batches of 16 utterances using Adam optimizer [20] and learning rate 0.001, which is halved every 5 epochs starting from the 32$^{\text{nd}}$ epoch. We use Polyak averaging with decay 0.998 [21]. All parameters are initialized using PyTorch defaults. We regularize with Gaussian weight noise with peak standard deviation $\sigma = 0.15$ [22], which increases linearly during initial $20k$ steps.

5. Experiments

We first confirm that locally normalized CTC with CD symbols conflicts with external language models. Figure 4 relates *Dev93* decoding accuracy with acoustic weight for global and local normalizations. For a given symbol, locally normalized CTC sharply differentiates between its contexts. This has to be mitigated during decoding with low acoustic model weights, which is equivalent to increasing the temperature of its SoftMax output classifier. Globally normalized CTC-G reaches a lower overall WER with higher acoustic weights, indicating that it produces more ambiguous hypotheses, which are resolved by the LM. We observed a similar trend for other locally and globally normalized models. In fact, for all CD models in Table 1 the optimal acoustic weight was close to 0.4 for locally normalized ones and to 1.0 for globally normalized ones.

[2]Our PyTorch implementation is freely available at `github.com/chorowski-lab/pytorch-asr`

Table 1: *WER (%) of character-based models on WSJ, with external LMs for locally normalized CTC, globally normalized CTC-G (e.g.* (2)*), CTC-GB with CD blanks. CDE denotes networks with the deep CD-symbol embedder. With the exception of tri-char models, we average 3 independent training runs.*

	Bg LM		Tg LM [9]	
Model	Dev93	Eval92	Dev93	Eval92
LF-MMI [4]				5.2*
LF-MMI Bi-char [4]				4.1*
Eesen [9]				7.3
Gram-CTC [23]				6.7
CTC/ASG [24]			9.5	6.6
CTC	12.1	8.7	9.3	6.6
CTC Bi-char	12.0	8.4	9.4	6.4
CTC Bi. CDE	11.9	8.4	9.3	6.4
CTC-G Bi-char	11.8	8.4	9.0	**6.2**
CTC-G Bi. CDE	11.6	8.7	9.0	6.5
CTC-GB Bi-char	11.8	8.5	9.2	6.5
CTC-GB Bi. CDE	11.5	8.5	**8.8**	**6.2**
CTC Tri-char	11.4	8.3	9.4	6.5
CTC Tri. CDE	**11.3**	**8.2**	8.9	6.4

* Uses additional data augmentation

We report in Table 1 error rates for CI and CD variants of CTC using the standard bigram (*Bg*) and trigram (*Tg*) [9] language models. Early stopping checkpoints and acoustic weights for *Eval92* have been selected by best WER on *Dev93*. We average the reported scores over 3 independent runs. As in previous sections, *CTC* denotes the typical, locally normalized variant (3), *CTC-G* the globally normalized one (2), and *CTC-GB* the globally normalized one with context-dependent blanks. Additionally, CDE denotes cases for which CD symbol prototypes were computed using the auxiliary embedding network instead of a look-up table of prototypes. Our baselines match the results of other CTC implementations [9, 23, 24]. For reference we provide LF-MMI results, however not directly comparable due to their data augmentation techniques.

The gains reported on the *Dev93* set, however small, are consistent and match the conclusions of our theoretical CTC loss analysis. Locally normalized bi-character CTC yields about 0.2 percentage point WER reduction, with slight gains coming from using CDE. Globally normalized CD CTC models reach the best performance, improving upon the baseline by 0.4 percentage points. However, the CDE module brings mixed results for globally normalized bi-character models. It yields a small reduction of the number of parameters, improves the error rates when contextual blanks are used, but seems inferior to a simple prototype look-up table under the typical CTC topology. This may indicate that a global CTC blank should have a representation that is unique and separate from other CD symbols, while the contextual blanks benefit from sharing of their prototypes.

The benefits of using the CDE module with tri-characters are more apparent. Not only it yields lower error rates, reaching the best WER with the bigram language model, but also the CDE module can handle tri-characters not seen during training, and requires substantially fewer parameters. In our case CDE has about $324k$ parameters, while regular embeddings of all $17k$ tri-characters allowed by the *Tg* language model would require as much as $5.5M$ parameters.

We attribute the poor performance of the tri-character CTC when decoded with the trigram LM, to overfitting the internal language model as described in Section 2.4. In fact, many of the triphones allowed by the trigram LM are not present in the training set, and the locally normalized CTC loss will make them improbable despite their prototypes being tied to other CD symbols by the CDE module, which yields only a small improvement over the CI baseline. Unfortunately, our forward-backward implementation does not allow computing the forward cost of the denominator graph in (2) with 17k states and we were not able to build globally normalized tri-character CTC models.

6. Related Work

CD symbols in neural ASR systems improve performance of CTC [3] and 1-state "chain" HMM models [4], with best results obtained with decision trees that map contexts to network targets. In [3] this tree is built from activations of the neural network, while [4] used a full bigram output which was slightly inferior to a decision tree obtained using a HMM model. The CDE module aims to recover the benefits of context-dependent outputs, but in a fully neural model that is trainable from scratch.

Several authors have proposed to use multicharacter, or multiphoneme output tokens to reduce the number of emissions. The DeepSpeech 2 model used non-overlappling character bigrams [2], while [23] and [25] dynamically chose a decomposition of the output sequence. This idea was also explored in [26] in the context of sequence-to-sequence models. We do not aim to reduce the length of target sequences, but to enable expressing of dependency of symbol emissions on their context.

Perhaps most similar to our work is [27] in which a neural network is trained to replace a state-tying decision tree. However, this implies a multistage training procedure in which both classical GMM-HMM and neural systems are built. In contrast, we strive to keep a simple, one stage training procedure.

Finally, Hypernetworks [28] expand on the idea of generating weights of neural network using an auxiliary module. We employ this idea in the CDE module and generate the prototypes for context-dependent symbols.

7. Conclusions

We have analyzed from a theoretical and experimental viewpoint the behavior of CTC with context-dependent targets. We have identified and addressed two major shortcomings: overfitting of locally normalized CTC, and the expansion of the number of parameters in the final layer of the network. Our proposed solutions are global normalization of the loss and dynamical computation of the final weight matrix using an auxiliary neural network, which allowed us to train compact networks with tri-character outputs. The changes preserve the simplified training procedure valued in the neural end-to-end systems, while yielding a 6% relative WER improvement on the WSJ dataset.

The scope for future work includes better integration with language models, scaling the global normalization to larger contexts, and experiments with LF-MMI chain models. We would also like to test the method on larger datasets.

8. Acknowledgments

The authors thank Polish National Science Center for funding under the Sonata 2014/15/D/ST6/04402 grant and to the PLGrid project for computational resources on the Prometheus cluster.

9. References

[1] Y. Hoshen, R. J. Weiss, and K. W. Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4624–4628.

[2] D. Amodei, S. Ananthanarayanan, and R. e. a. Anubhai, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 173–182.

[3] A. Senior, H. Sak, and I. Shafran, "Context dependent phone models for lstm rnn acoustic modelling," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4585–4589.

[4] H. Hadian, H. Sameti, D. Povey, and S. Khudanpur, "End-to-end speech recognition using Lattice-free MMI," in *Proceedings of Interspeech*, 2018, pp. 12–16.

[5] 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*, 2006, pp. 369–376.

[6] L. 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.

[7] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Found. Trends Signal Process.*, vol. 1, no. 3, pp. 195–304, 2007.

[8] S. J. Young, J. J. Odell, and P. C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proceedings of the Workshop on Human Language Technology*, 1994, pp. 307–312.

[9] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding." in *Proceedings of ASRU*, 2015, pp. 167–174.

[10] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI," Sep. 2016, pp. 2751–2755.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an End-to-End ConvNet-based Speech Recognition System," *arXiv:1609.03193 [cs]*, Sep. 2016, arXiv: 1609.03193.

[13] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-Based Models for Speech Recognition," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 577–585.

[14] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[15] M. Mohri, F. Pereira, and M. Riley, "Speech Recognition with Weighted Finite-State Transducers," in *Springer Handbook of Speech Processing*, P. J. B. Dr, P. M. M. Sondhi, and P. Y. A. H. Dr, Eds. Springer Berlin Heidelberg, Jan. 2008, pp. 559–584.

[16] J. Bridle and L. Dodd, "An alphanet approach to optimising input transformations for continuous speech recognition," in *Proceedings of the Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference*. IEEE Computer Society, 1991, pp. 277–280.

[17] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The Kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

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

[21] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[22] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 2348–2356.

[23] H. Liu, Z. Zhu, X. Li, and S. Satheesh, "Gram-CTC: Automatic unit selection and target decomposition for sequence labelling," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2188–2197.

[24] N. Zeghidour, N. Usunier, G. Synnaeve, R. Collobert, and E. Dupoux, "End-to-end speech recognition from the raw waveform," in *Interspeech 2018*, 2018.

[25] O. Siohan, "CTC training of multi-phone acoustic models for speech recognition," in *Proceedings of Interspeech*, 2017, pp. 709–713.

[26] W. Chan, Y. Zhang, Q. Le, and N. Jaitly, "Latent sequence decompositions," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[27] M. Yadav and V. Tyagi, "Deep triphone embedding improves phoneme recognition," *arXiv preprint arXiv:1710.07868*, 2017.

[28] D. Ha, A. Dai, and Q. Le, "Hypernetworks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.