# Regarding Topology and Variant Frame Rates for Differentiable WFST-based End-to-End ASR

*Zeyu Zhao, Peter Bell*

Centre for Speech Technology Research, University of Edinburgh, UK

zeyu.zhao@ed.ac.uk, peter.bell@ed.ac.uk

## Abstract

End-to-end (E2E) Automatic Speech Recognition (ASR) has gained popularity in recent years, with most research focusing on designing novel neural network architectures, speech representations, and loss functions. However, the importance of topology in E2E ASR has been largely neglected. There are many aspects of topology to consider; in this paper, we focus on the relationship between topologies' minimum traversal time and output frame rate, the number of distinct states for each output unit, and the flexibility of alignments admitted. We examine several different topologies on two datasets: WSJ and Librispeech. Our experiments reveal that different frame rates have varying optimal topologies and that the commonly used Connectionist Temporal Classification (CTC) topology is not always optimal. Our findings suggest that the choice of topology is an important consideration in the design of E2E ASR systems.

**Index Terms**: Automatic Speech Recognition, End-to-End ASR, Differentiable WFST

## 1. Introduction

Automatic speech recognition (ASR) is the task of transcribing input speech into the corresponding word sequence as accurately as possible. One of the main challenges in ASR is dealing with the absence of alignment information, where there is speech data and the corresponding sentence-level transcription [1]. Conventional ASR methods, such as Hidden Markov Model (HMM) and Baum-Welch algorithm, tackle the alignment issue by considering all possible alignments [1]. On the other hand, end-to-end (E2E) ASR models, such as Connectionist Temporal Classification (CTC) [2], Recurrent Neural Network Transducer (RNN-T) [3], and Lattice Free Maximum Mutual Information (LF-MMI) [4, 5], model the posterior probability of the target transcription by summing the probabilities of all possible alignments. The Attention-based Encoder-Decoder (AED) framework also deals with alignments using an attention mechanism [6, 7].

The above direction of E2E ASR research is to address the absence of alignment information by computing the objective function and backpropagating the loss for training deep neural network (DNN) models. Another direction is to design new DNN architectures to improve E2E ASR performance. In recent years, many types of DNN architectures have been proposed, such as Transformer [8] and Conformer [9], which significantly improve performance. In addition to designing new DNN architectures, researchers have also focused on self-supervised or unsupervised learning to fully utilize unlabelled data [10, 11, 12, 13]. These methods make full use of unlabelled data to extract better speech representations [10, 11], or to generate pseudo-labels [12, 13].

While research on E2E ASR has mainly focused on loss functions, neural network architectures, and self-supervised learning, there has been less emphasis on the (often implicit) role of the model *topology*. In contrast, in conventional HMM-based ASR, HMM topology was subject to detailed investigations in the past [14, 15, 16, 17]. Topology, in this context, refers to how the outputs of acoustic models (in our case, DNNs) – or tokens – related to the output units, which can be characters, phonemes, or Byte-Pair Encoding (BPE) units. Whilst attention-based encoder-decoder (AED) models operate in a purely continuous space, the widely CTC and RNN-T models do have an HMM-like topology, and it should be noted that CTC plays a key role in regularising AED training alignments in many systems. CTC models can be considered to have a one-state-HMM topology with a single self-loop, and as such, the importance of topology in such models is generally overlooked [18]. However, recent works have demonstrated that topology is a crucial aspect of ASR models [18, 19, 20]. [20] explored the relationship between topology and alignments and found that altering the topology can affect the alignment density and that certain alignments can dominate over others, leading to improved alignment quality. Similarly, in [19], the authors compared the performance of different topologies within the LF-MMI framework. Despite the importance of topology, many aspects of its influence on modern ASR models remain to be investigated in a precise and systematic way. One potential challenge that researchers faced when working with topology in ASR is the implementation, particularly when combining with deep learning frameworks. However, the Differentiable Weighted Finite State Transducer (DWFST) offers a flexible approach to integrating WFST formalism with end-to-end (E2E) deep learning frameworks, enabling researchers to investigate the impact of topology on ASR models [21]. In this paper, we focus on the interaction between topology and output frame rates in the DWFST E2E ASR framework. Specifically, we examine three main aspects: (1) the impact of output frame rate on the optimal topology selection, particularly in respect of its minimum traversal time (2) the effect of topology and output frame rate on alignment flexibility, and (3) the influence of the output frame rate and number of states in the topology on the behaviour of the shared blank label.

## 2. Methods

### 2.1. Training

We begin by obtaining a sequence of hidden features, or high-level representations, $H = h_{1:T}$, from the input speech features sequence, $X = x_{1:T}$, using an acoustic encoder. The acoustic encoder may optionally apply subsampling, resulting in a hidden feature sequence of length smaller than that of the input sequence. For simplicity, we use $T$ to denote both sequence lengths of $X$ and $H$. Next, we apply a linear layer with softmax activation to obtain a probability distribution over all tokens, $p_t(c|X)$, where $t = 1, 2, \ldots, T$ and $c \in A$ is a token from the token set $A$. We assume conditional independence, like CTC, so that the probability of a token sequence $\pi = \pi_{1:T}, \pi_t \in A$ can be written as

$$p(\pi|X) = \prod_{t=1}^{T} p_t(\pi_t|X), \tag{1}$$

Then, the posterior probability, $p(Y|X)$, for a given word sequence $Y = y_{1:U}$ is:

$$p(Y|X) = \frac{\sum_{\pi:Y} p(\pi|X)}{\sum_{Y'} \sum_{\pi:Y'} p(\pi|X)}, \tag{2}$$

where the summation over $Y'$ is performed over all possible word sequences, and $\pi : Y$ denotes all the paths corresponding to the word sequence $Y$. We use the training graph $S^{\mathrm{TRN}}$ to compute the numerator by

$$\sum_{\pi:Y} p(\pi|X) = \mathrm{TotalScore}(\mathrm{Intersect}(E, S^{\mathrm{TRN}})), \tag{3}$$

where Intersect and TotalScore represent the intersection and total score in log semiring, respectively, $E$ is the Emission FST transformed from the output of the neural network model, and $S^{\mathrm{TRN}}$ is the training graph

$$S^{\mathrm{TRN}} = T \circ (L \circ W), \tag{4}$$

where $T$, $L$, and $W$ are the Token, the Lexicon and the Transcription FST, respectively [21]. To calculate the denominator, we only consider the valid paths, and thus

$$\sum_{Y'} \sum_{\pi:Y'} p(\pi|X) = \mathrm{TotalScore}(E \circ T), \tag{5}$$

where $E$ is the emission FST. Note that with most topologies (except CTC topology), not all the paths in $E$ are valid, which makes the summation of the probabilities of all possible word sequences not equal to one. A previous work [22] has shown that the normalisation (the denominator term in eq. (2)) is crucial for the sequence-level loss function. Finally, our objective function is defined as:

$$\mathcal{L} = -\log p(Y|X) \tag{6}$$

### 2.2. Decoding

The decoding graph is constructed as follows:

$$S^{\mathrm{DEC}} = T \circ (L \circ G), \tag{7}$$

where $G$ is a grammar FST, often obtained from an n-gram language model [23]. In this equation, we have ignored many operations, such as determinization and minimization, for simplicity [24]. Given the output of the DNN model and the decoding graph, we apply a Viterbi decoder to find the best path [25]. The decoding result, $W^*$, is

$$W^* = \arg\max_W (\log p(W) + \alpha \max_{\pi:W} \log p(\pi|X)), \tag{8}$$

where $\pi : W$ denotes all the paths that correspond to the word sequence, $W$, considering the decoding graph, $S^{\mathrm{DEC}}$.

## 3. Experiments and Analysis

### 3.1. Topologies

There are eight different topologies examined in this paper as shown in table 1. The first topology, S1-T1, is the standard CTC topology, which serves as a baseline for our experiments. In CTC topology, there is one single token for each phone and a shared blank label that is mandatory between repeated neighbouring phones as a separator. Note that in [26], the authors compared different CTC topologies, while in this work, we compare different topologies most of which are not equivalent to CTC. Inspired by previous work [22], we introduce an extra state for each phone to increase the modelling power, resulting in the S2-T1 topology in table 1, where there is no self-loop for the first state in S2-T1, and the second state is optional and skippable. S2-T1★ is similar to S2-T1, except for the self-loop on the first state. Interestingly, we introduce S2-T1★ because all the alignments acceptable by S2-T1 can be accepted by S2-T1★. In this way, we can investigate how the number of possible alignments (or the size of the alignment space) affects model performance.

All of the above three topologies can be traversed by one output frame. To further investigate, we introduce five more topologies that absorb at least two frames at the output end. S2-T2★ is a direct generalization of S1-T1 by adding an extra state with a self-loop. Similarly to S2-T1 and S2-T1★, S2-T2 is a variant of S2-T2★ without self-loop. Keeping the same minimum traversal frame but adding one extra state to investigate whether one more state can boost the modelling power, we have S3-T2 and its variants, S3-T2★ and S3-T2★★.

In all the above topologies, we keep the shared optional blank token to ensure a fair comparison with the original CTC and to investigate how the topology and the output frame affect its behaviour. In other words, the only difference between CTC and the other topologies is how many tokens we use to model a phone and what the topology (whether the self-loop or skip transition is allowed or not) is.

### 3.2. Basic Settings

We apply Kaldi [27] and ESPnet [28] for data preparation, PyTorch [29] for DNN training and k2[1] as the DWFST backend. To facilitate reproducibility and further research on topology, we make our code open source[2].

Our experiments are conducted on the WSJ and Librispeech datasets. For both datasets, we extract 80-dimensional Fbank features and 3-dimensional pitch features, perform speaker-wise CMVN, and use the resulting features as inputs to our neural network models. We further improve the performance by applying SpecAugmentation [30] to the input features. As for output tokens (*phones* in our framework), we select Byte Pair Encoding (BPE) [31] for both datasets. Namely, we extract 100 and 5000 BPEs for WSJ and Librispeech, respectively. To investigate how the topology interacts with the frame rate, for each dataset, we apply two different subsampling factors in the NN models by controlling the input convolution layers. Specifically, for WSJ, the two subsampling factors of 2 and 4, but 4 and 6 for Librispeech. The reason why we choose these two specific frame rates for WSJ and Librispeech can be briefly explained by table 2, where the average ratio of the NN output and the target (BPEs) sequence is shown. We can see that to

[1]https://github.com/k2-fsa/k2
[2]https://github.com/ZhaoZeyu1995/Waterfall

Table 1: *Topologies comparison where 'blk' denotes the shared optional blank label. Note that sometimes the blank label is unskippable for CTC but we ignore it in this table for simplicity. $Sx$-$Ty$ means there are $x$ states for each phone and the minimum traversal frame is $y$ and one ★ means there is one more self-loop added.*

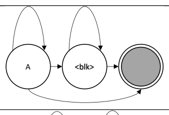
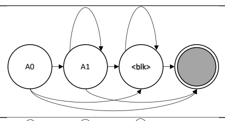
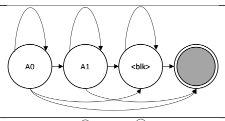
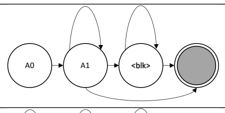
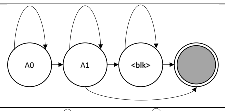
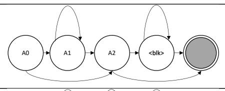
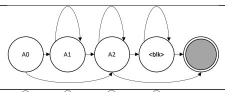
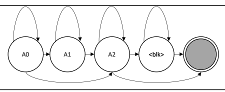
| Topology | Figure | Description | #States | Minimum Traversal |
|---|---|---|---|---|
| S1-T1 (CTC) |  | One state with self-loop | 1 | 1 |
| S2-T1 |  | No self-loop on the first state and skippable second state | 2 | 1 |
| S2-T1★ |  | Skippable second state | 2 | 1 |
| S2-T2 |  | No self-loop on the first state not skippable | 2 | 2 |
| S2-T2★ |  | Two states with self-loop not skippable | 2 | 2 |
| S3-T2 |  | No self-loop on the first and the third state with a skippable second state | 3 | 2 |
| S3-T2★ |  | No self-loop on the first state with a skippable second state | 3 | 2 |
| S3-T2★★ |  | A skippable second state | 3 | 2 |

Table 2: *The average ratio of the output and the target length with different subsampling factors.*

| | 2 | 4 | 6 | 8 |
|---|---|---|---|---|
| WSJ | 6.33 | 3.16 | 2.10 | 1.57 |
| Librispeech | 15.40 | 7.69 | 5.12 | 3.83 |

investigate the topology with a minimum traversal frame of 2, for WSJ, we have to apply subsampling factors of 4 or 2. Even though 6 might be still feasible for WSJ theoretically, the alignment space is too small limiting the ability of the model to learn the alignment. For Librispeech, in our primary experiments, we found that the performance with a subsampling factor of 8 is really bad. Considering this issue, and to keep a similar average ratio between the length of the output and the target sequences, we finally choose subsampling factors of 4 and 6 for Librispeech. We suppose the reason why the performance with a subsampling factor of 8 is significantly bad, is that the number of BPEs for Librispeech is 5000, meaning each BPE on average represents a longer context and needs to absorb more frames than 100 BPEs for WSJ.

We apply transformer [8] and conformed [32] as our end-to-end ASR model for WSJ and Librispeech, respectively. Our models are trained by Adam optimizer [33] with a linear warmup learning rate scheduler and the hyperparameters are the same as [9].

## 3.3. WSJ

In addition to SpecAugmentation, we also apply speed perturbation to obtain the speech data with 0.9x and 1.1x of the original speed, resulting in approximately three times the original data.

We first train a baseline CTC model with our framework and compare it with different models based on ESPnet as shown in table 3. In comparison with pure CTC or AED without language models, our baseline outperforms them, indicating that our framework is set up correctly. table 4 and table 5 show the results obtained with different subsampling factors, topologies and language models. We evaluate our models with the official 3-gram language model and the 4-gram language model trained with the conventional Kaldi recipe. To assess the impact of topology and output frame rate on the behaviour of the blank label, we calculate the ratio of the blank label being the maximum in the model outputs on eval92, the "Blank Ratio" column in tables 4 and 5.

Table 3: *Baseline performance (WER%) compared with other ESPnet models on WSJ*

| | dev93 | eval92 |
|---|---|---|
| CTC baseline w/o LM (ours) | 15.6 | 12.7 |
| ESPnet CTC w/o LM [9] | N/A | 15.5 |
| ESPnet AED w/o LM [3] | 18.1 | 14.0 |

Table 4: *The performance (WER%) and the Blank Ratio (%) of different models with a subsampling factor (SF) of 4 on WSJ*

| SF=4 | 4-gram LM | | 3-gram LM | | Blank Ratio |
|---|---|---|---|---|---|
| | dev93 | eval92 | dev93 | eval92 | |
| S1-T1 (CTC) | 12.9 | 9.5 | 14.5 | 11.2 | 46.65 |
| S2-T1 | **8.8** | **6.1** | **11.4** | **8.3** | 23.52 |
| S2-T1★ | 9.0 | 6.2 | 11.6 | 8.4 | 22.94 |
| S2-T2★ | 13.7 | 9.7 | 15.8 | 11.5 | 0.84 |
| S2-T2 | 12.8 | 9.9 | 15.4 | 12.1 | 1.69 |
| S3-T2 | 14.0 | 10.3 | 16.0 | 11.9 | 2.19 |
| S3-T2★ | 13.6 | 9.6 | 15.6 | 11.4 | 1.89 |
| S3-T2★★ | 13.9 | 10.3 | 15.9 | 11.8 | 1.76 |

Table 5: *The performance (WER%) of different models with a subsampling factor (SF) of 2 on WSJ*

| SF=2 | 4-gram LM | | 3-gram LM | | Blank Ratio |
|---|---|---|---|---|---|
| | dev93 | eval92 | dev93 | eval92 | |
| S1-T1 (CTC) | 10.2 | 7.7 | 12.3 | 9.4 | 73.44 |
| S2-T1 | 10.6 | 7.1 | 13.3 | 9.6 | 24.31 |
| S2-T1★ | 9.5 | 6.5 | 12.2 | 9.0 | 24.37 |
| S2-T2★ | **9.1** | **6.3** | **11.8** | **8.6** | 25.79 |
| S2-T2 | 11.9 | 7.5 | 15.4 | 10.1 | 22.73 |
| S3-T2 | 10.8 | 6.8 | 13.4 | 9.4 | 0.14 |
| S3-T2★ | 10.4 | 6.4 | 13.0 | 9.1 | 0.23 |
| S3-T2★★ | 10.9 | 6.9 | 13.5 | 9.6 | 0.19 |

### 3.4. Librispeech

We train Conformer models [32] on train960 from scratch. We set the number of transformer blocks to 12, each with a hidden size of 512 and 8 attention heads. The feed-forward network has an inner dimension of 2048. The language models we apply here are the official 3-gram (tglarge) and 4-gram (fglarge) language models provided by the dataset and blank ratios are calculated on test-clean.

Table 6: *The performance (WER%) of different topologies with a subsampling factor of 6 on Librispeech*

| Sub6 | 4-gram LM | | 3-gram LM | | Blank Ratio |
|---|---|---|---|---|---|
| | test-clean | test-other | test-clean | test-other | |
| S1-T1(CTC) | 6.1 | 14.9 | 6.4 | 15.4 | 47.58 |
| S2-T1 | **4.8** | **13.4** | **5.1** | **13.8** | 24.56 |
| S2-T1★ | 4.9 | 13.7 | 5.4 | 14.1 | 24.32 |
| S2-T2★ | 6.3 | 15.4 | 6.7 | 15.8 | 1.12 |
| S2-T2 | 6.5 | 15.7 | 6.8 | 16.0 | 1.76 |

Table 7: *The performance (WER%) of different topologies with a subsampling factor of 4 on Librispeech*

| Sub4 | 4-gram LM | | 3-gram LM | | Blank Ratio |
|---|---|---|---|---|---|
| | test-clean | test-other | test-clean | test-other | |
| S1-T1(CTC) | 5.9 | 14.6 | 6.1 | 15.9 | 75.36 |
| S2-T1 | 6.2 | 14.8 | 6.5 | 16.0 | 23.22 |
| S2-T1★ | 5.2 | 14.2 | 5.5 | 15.3 | 23.76 |
| S2-T2★ | **4.9** | **13.5** | **5.1** | **14.7** | 24.37 |
| S2-T2 | 6.5 | 15.2 | 6.8 | 15.9 | 22.47 |

### 3.5. Analysis

With a fixed subsampling factor, we observe that S2-T1 outperforms other topologies for larger subsampling factors (4 for WSJ and 6 for Librispeech), as shown in tables 4 and 6, and topologies *-T2* perform worse. This is because a larger subsampling factor results in a shorter average output sequence length, which means there are fewer possible alignments for topologies with a #Minimum Traversal of 2, as they have longer target sequences at the token level. This is analogous to an extreme case where the length of the target sequence always matches that of the output sequence, resulting in only one possible alignment and training the model using a cross-entropy loss. However, the model may experience confusion if acoustic features with the same label are not easily classified to be the same class. Note that the alignment space of S2-T1 is a subset of S2-T1★, which means all the alignments acceptable by the former can always be accepted by the latter. However, a larger alignment space does not necessarily result in better performance. Comparing S3-T2, S3-T2★, and S3-T2★★, considering adding a self-loop makes the alignment space larger, we find that adding one self-loop can make the performance better but worse with two. Besides, as shown in tables 5 and 7, when we decrease the subsampling factor for both datasets, the best performance is achieved by S2-T2★, but not S2-T1. Decreasing the subsampling factor increases the alignment space for all topologies, and we find that the performance improves for most topologies, except S2-T1 and S2-T1★. Furthermore, although the alignment space of CTC, S2-T1, and S2-T1★ is larger than that of S2-T2★, our results show that the S2-T2★ achieves the best performance. This suggests that having too large an alignment space makes it harder for a model to train effectively. Therefore, it is crucial to choose a proper topology that reflects the acoustic characteristics of the input data and to consider the Minimum Traversal value based on the subsampling factor when training a model.

Among topologies *-T1*, we find that S2-T1 performs the best compared to CTC and S2-T1★. Interestingly, S2-T1★ is slightly worse than S2-T1 but better than CTC, indicating that the modelling ability increases as we introduce a new token for each phone in the topology. However, comparing S2-T2, S2-T2★, S3-T2 and its variants, we find that adding new tokens does not necessarily improves performance, as it introduces more classes and makes it harder to classify.

Our study also sheds light on the behaviour of the blank label in relation to the topology and output frame rate. We found that with a larger subsampling factor, the blank ratio is primarily related to the T value (#Minimum Traversal) of the topology, as shown in tables 4 and 6. Conversely, with a smaller subsampling factor, the blank ratio is mostly related to the S value (#States), as demonstrated in tables 5 and 7. The blank label ratio is crucial for alignment quality [18], and if the majority of frames are classified as blank labels, the model outputs become excessively peaky, and the model's alignments become less reliable. Therefore, topology selection plays a critical role in controlling the blank ratio. Specifically, with a smaller subsampling rate, we should prioritise #States, and with a larger subsampling rate, we should focus on #Minimum Traversal.

## 4. Conclusions

In this paper, we systematically investigated the impact of different topologies and output frame rates on the performance of a DWFST-based E2E ASR. Our findings demonstrate the importance of carefully selecting a topology in order to effectively train a model. We also discussed how to choose topology to have more control of the blank ratio. These results can inform the development of future E2E ASR systems, leading to improved performance and more accurate transcription of speech data.

## 5. References

[1] M. Gales and S. Young, “The Application of Hidden Markov Models in Speech Recognition,” *Foundations and Trends® in Signal Processing*, vol. 1, no. 3, pp. 195–304, 2007.

[2] A. Graves, S. Fernández *et al.*, “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.

[3] A. Graves, “Sequence Transduction with Recurrent Neural Networks,” *arXiv:1211.3711 [cs, stat]*, Nov. 2012.

[4] H. Hadian, H. Sameti *et al.*, “End-to-end Speech Recognition Using Lattice-free MMI,” in *Interspeech 2018*. ISCA, Sep. 2018, pp. 12–16.

[5] D. Povey, V. Peddinti *et al.*, “Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI,” in *Interspeech 2016*. ISCA, Sep. 2016, pp. 2751–2755.

[6] W. Chan, N. Jaitly *et al.*, “Listen, Attend and Spell,” *arXiv:1508.01211 [cs, stat]*, Aug. 2015.

[7] J. Chorowski, D. Bahdanau *et al.*, “Attention-Based Models for Speech Recognition,” *arXiv:1506.07503 [cs, stat]*, Jun. 2015.

[8] A. Vaswani, N. Shazeer *et al.*, “Attention Is All You Need,” *arXiv:1706.03762 [cs]*, Dec. 2017.

[9] P. Guo, F. Boyer *et al.*, “Recent Developments on ESPnet Toolkit Boosted by Conformer,” Oct. 2020.

[10] A. Baevski, H. Zhou *et al.*, “Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” *arXiv:2006.11477 [cs, eess]*, Oct. 2020.

[11] W.-N. Hsu, B. Bolte *et al.*, “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units,” Jun. 2021.

[12] S. Ling, C. Shen *et al.*, “Improving Pseudo-Label Training For End-To-End Speech Recognition Using Gradient Mask,” in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8397–8401.

[13] Y. Higuchi, N. Moritz *et al.*, “Advancing Momentum Pseudo-Labeling with Conformer and Initialization Strategy,” in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7672–7676.

[14] R. Bakis, “Continuous speech recognition via centisecond acoustic states,” *The Journal of the Acoustical Society of America*, vol. 59, no. S1, pp. S97–S97, 1976.

[15] L. Rabiner and B. Juang, “An introduction to hidden Markov models,” *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, Jan. 1986, conference Name: IEEE ASSP Magazine.

[16] L. R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” *PROCEEDINGS OF THE IEEE*, vol. 77, no. 2, p. 30, 1989.

[17] B. H. Juang and L. R. Rabiner, “Hidden Markov Models for Speech Recognition,” *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991, publisher: [Taylor & Francis, Ltd., American Statistical Association, American Society for Quality]. [Online]. Available: https://www.jstor.org/stable/1268779

[18] A. Zeyer, E. Beck, R. Schlüter, and H. Ney, “CTC in the Context of Generalized Full-Sum HMM Training,” in *Interspeech 2017*. ISCA, Aug. 2017, pp. 944–948.

[19] H. Hadian, H. Sameti *et al.*, “Flat-Start Single-Stage Discriminatively Trained HMM-Based Models for ASR,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 1949–1961, 2018.

[20] T. Raissi, W. Zhou, S. Berger, R. Schlüter, and H. Ney, “HMM vs. CTC for Automatic Speech Recognition: Comparison Based on Full-Sum Training from Scratch,” in *2022 IEEE Spoken Language Technology Workshop (SLT)*, Jan. 2023, pp. 287–294.

[21] A. Hannun, V. Pratap *et al.*, “Differentiable Weighted Finite-State Transducers,” *arXiv:2010.01003 [cs, stat]*, Oct. 2020.

[22] Z. Zhao and P. Bell, “Investigating Sequence-Level Normalisation For CTC-Like End-to-End ASR,” in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 7792–7796.

[23] Y. Miao, M. Gowayyed, and F. Metze, “EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding,” in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec. 2015, pp. 167–174.

[24] M. Mohri, F. Pereira, and M. Riley, “Speech Recognition with Weighted Finite-State Transducers,” in *Springer Handbook of Speech Processing*, J. Benesty *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 559–584.

[25] D. Povey, M. Hannemann *et al.*, “Generating exact lattices in the WFST framework,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2012, pp. 4213–4216.

[26] A. Laptev, S. Majumdar, and B. Ginsburg, “CTC Variations Through New WFST Topologies,” in *Interspeech 2022*. ISCA, Sep. 2022, pp. 1041–1045.

[27] D. Povey, A. Ghoshal *et al.*, “The Kaldi speech recognition toolkit,” in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, 2011.

[28] S. Watanabe, T. Hori *et al.*, “ESPnet: End-to-End Speech Processing Toolkit,” *arXiv:1804.00015 [cs]*, Mar. 2018.

[29] A. Paszke, S. Gross *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[30] D. S. Park, W. Chan *et al.*, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” *Interspeech 2019*, pp. 2613–2617, Sep. 2019.

[31] 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 (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725.

[32] A. Gulati, J. Qin *et al.*, “Conformer: Convolution-augmented Transformer for Speech Recognition,” May 2020.

[33] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” https://arxiv.org/abs/1412.6980v9, Dec. 2014.