Phone Synchronous Speech Recognition With CTC Lattices

Zhehuai Chen, *Student Member, IEEE*, Yimeng Zhuang, Yanmin Qian, *Member, IEEE*, and Kai Yu, *Senior Member, IEEE*

***Abstract*—Connectionist temporal classification (CTC) has recently shown improved performance and efficiency in automatic speech recognition. One popular decoding implementation is to use a CTC model to predict the phone posteriors at each frame and then perform Viterbi beam search on a modified WFST network. This is still within the traditional frame synchronous decoding framework. In this paper, the peaky posterior property of CTC is carefully investigated and it is found that ignoring blank frames will not introduce additional search errors. Based on this phenomenon, a novel *phone synchronous decoding* framework is proposed by removing tremendous search redundancy due to blank frames, which results in significant search speed up. The framework naturally leads to an extremely compact phone-level acoustic space representation: *CTC lattice*. With CTC lattice, efficient and effective modular speech recognition approaches, second pass rescoring for large vocabulary continuous speech recognition (LVCSR), and phone-based keyword spotting (KWS), are also proposed in this paper. Experiments showed that phone synchronous decoding can achieve 3–4 times search speed up without performance degradation compared to frame synchronous decoding. Modular LVCSR with CTC lattice can achieve further WER improvement. KWS with CTC lattice not only achieved significant equal error rate improvement, but also greatly reduced the KWS model size and increased the search speed.**

***Index Terms*—CTC, Decoder, DLSS, KWS, Lattice, LVCSR, WFST.**

I. Introduction

AUTOMATIC speech recognition (ASR) is both a pattern recognition and search problem [1]. Speech recognition errors come from both sides, called modeling error and search error respectively. The search process of a speech recognizer is to find a sequence of labels whose corresponding acoustic and language models best match the input feature so as to minimize the search error. According to different modeling units between acoustic and language models, such kind of search algorithms can mainly be divided into two types, breadth-first style *time synchronous beam search* and depth-first style *stack decoding*(A* search). As the heuristic function for A* search is hard to be obtained in large vocabulary continuous speech recognition (LVCSR) [2] and not suitable for online speech recognition application, beam search approach has been dominant for decades. The approach operates at each frame using Viterbi algorithm with beam pruning [3], hence referred to as *frame synchronous* search. Despite the wide adoption, the approach has some weakness. Firstly, it is an equal interval search algorithm and is inefficient to deal with the dynamic changes of time gap between acoustic and linguistic units (e.g. phones). Secondly, it is usually hard to balance search efficiency and search errors for LVCSR.

For the first problem, *frame skipping* (FS) [4] [5], or more sophisticated *variable frame rate* (VFR) schemes [6], [7] are proposed to achieve unequal search interval. However, such methods only focus on acoustic features without considering the granularity gaps between acoustic frame and linguistic units. Therefore, these feature-level variable frame rate methods can not dynamically adjust the search interval to match the information rate of linguistic knowledge sources. This is the main limitation of the prior arts.

As for the second problem, two decoding frameworks have been investigated to address it.

1) *Integrated search with search space optimization:* Weighted finite state transducer (WFST) [8] is proposed to offline combine different knowledge sources (acoustic and language models, etc) and perform search space optimization to achieve best searching efficiency in advance [9], [10]. The history conditioned lexical tree search [11] is another strategy for integrating different knowledge sources but by dynamic expansion. Meanwhile, runtime pruning is conducted via the Viterbi beam search process on all decoding paths [12] with sophisticated lookahead strategies [13], to improve search efficiency. However, integrating all knowledge sources together leads to a huge search space. In real time decoder, search error becomes even more critical because of harder pruning to reduce exhaustive search.
2) *Multi-pass modular search:* In [14], HMM-state is generated for each frame of the whole utterance and mapped to a phone lattice. Then other knowledge sources (e.g., phonemic error model [15], keyword sequence [16] or language model [17]) are applied to the lattice using dynamic programming or WFST composition. However, the time complexity of the method is large while performance enhancement is minor. The key reason is that acoustic information

Manuscript received August 14, 2016; revised November 1, 2016; accepted November 2, 2016. Date of publication November 4, 2016; date of current version November 28, 2016. This work was supported in part by the Shanghai Sailing Program under Grant 16YF1405300, in part by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, in part by the China NSFC projects under Grants 61573241 and 61603252, and in part by the Interdisciplinary Program (14JCZ03) of Shanghai Jiao Tong University in China. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sabato Marco Siniscalchi. *(Corresponding authors: Kai Yu and Yanmin Qian.)*

The authors are with the Computer Science and Engineering Department, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: chenzhehuai@sjtu.edu.cn; yimmon@sjtu.edu.cn; yanminqian@sjtu.edu.cn; kai.yu@sjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2016.2625459

in HMM is not concentrated (HMM state loops and last for several frames) and the model unit is smaller than a phone. This results in large phone lattice and makes lattice pruning very hard to trade off between accuracy and speed.

Connectionist temporal classification (CTC) [18] has been proposed as a new type of acoustic model in ASR and achieved state-of-the-art performance [19], [20], [21], [22]. Besides, context independent monophone CTC model also shows competitive performance compared with context dependent state clustered hybrid neural network HMM model [22], [23], [24], [25]. CTC model usually outputs a highly peaky distribution. An interesting characteristics is that the majority of frames have the `blank` symbol as the most likely output [23]. Hence, there exist long spans of `blank` in the search space. Within these `blank` spans, since linguistic information (i.e. phones or words) does not change at all, any search effort, i.e. acoustic and linguistic score combination and token passing, becomes redundant. When a phone starts dominating the distribution, linguistic search space changes. Only at this point should search be continued. We name this characteristics as *discontinuous linguistic search space* (DLSS) phenomenon of CTC-trained model. It is further analyzed in Section III-A.

In this paper, the potential of the DLSS phenomenon in CTC-trained model is utilized to remove tremendous search redundancy due to `blank` frames. In this way, frame synchronous decoding (FSD) is transformed into *phone synchronous decoding* (PSD) with self-adjusted interval in WFST-based decoding. Such process can also be viewed as a variant of variable frame rate analysis. However, in contrast to all prior arts, PSD frame rate analysis is carried out at model level rather than feature level, which results in higher efficiency. The PSD framework naturally leads to the construction of phone-level *CTC lattice*. CTC lattice is extremely compact and preserves almost all acoustic information. Since phone lattice is the basis for modular search paradigm of ASR, phone-level CTC lattice benefits various modular search approaches. Two CTC lattice motivated approaches, LVCSR rescoring and keyword spotting, are investigated in this paper.

The rest of the paper is organized as follows. In section II, CTC model training is briefly reviewed. In section III, phone synchronous decoding and CTC lattice are introduced in detail. Integrated and modular search methods with CTC lattice are discussed in section IV. Section V describes experiments and analysis, followed by the conclusion in section VI.

II. CTC Acoustic Model

In this section, *Connectionist temporal classification* (CTC) model [18] and its characteristics are briefly introduced. Originally, CTC model is proposed with bi-directional RNN. Since it is challenging to deploy bi-directional RNN models in an online, low-latency decoding setting [21], this paper mainly focuses on uni-directional LSTM based CTC model. It is noted that all conclusions in this paper also apply to bi-directional RNN based CTC models.

A CTC model predicts the conditional probability of a label sequence (phone sequence in ASR) by summing over the joint probabilities of the corresponding set of CTC symbol sequences (called *paths*). Assuming CTC symbols are conditionally independent at each frame, the conditional probability of the whole label sequence is given by

$$P(\mathbf{l}|\mathbf{x}) = \sum_{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\mathbf{l})} P(\boldsymbol{\pi}|\mathbf{x}) = \sum_{\pi:\pi\in L',\mathcal{B}(\boldsymbol{\pi}_{1:T})=\mathbf{l}} \prod_{t=1}^{T} y_{\pi_t}^t \quad (1)$$

where , $\mathbf{l} = (l_1, \ldots, l_U)$ denotes a phone label sequence consisting U phones, $l \in L$ and L is the phone set for ASR. $\mathbf{x} = (x_1, \ldots, x_T)$ is the corresponding feature sequence of T frames, t is the index of frame and T is the total number of frames. In CTC, it is assumed that $U \leq T$. $\boldsymbol{\pi}_{1:T} = (\pi_1, \ldots, \pi_T)$ is the frame-wise decoding *path* from frame 1 to T, i.e. CTC output symbol sequence. Each output symbol $\pi \in L'$ and $L' = L \cup \{\texttt{blank}\}$. `blank` is a special model unit introduced within the CTC framework. It represents a label of "null" and models the acoustic variabilities outside the defined phone sets. y_k^t is the probability of output symbol of CTC network k at time t. A many-to-one mapping $\mathcal{B}$ is defined as $\mathcal{B}: L' \mapsto L$ to determine the correspondence between a set of *paths* and a phone label sequence. The mapping rule is to first remove the repeated phone labels and then all `blank` symbols from the paths.

The CTC objective function $\mathcal{J}$ is defined as the negative log conditional probability of the correct labels of all training sequences

$$\mathcal{J} = -\sum_{n=1}^{N} \ln(P(\mathbf{l}_n|\mathbf{x}_n)) \quad (2)$$

where n is the index of training sequences. The gradient of (2) with respect to (w.r.t.) the output at each frame can be calculated as (using one training sequence as an example)

$$\frac{\partial \mathcal{J}}{\partial y_k^t} = -\frac{\partial \ln(P(\mathbf{l}|\mathbf{x}))}{\partial y_k^t} = -\frac{1}{P(\mathbf{l}|\mathbf{x})}\frac{\partial P(\mathbf{l}|\mathbf{x})}{\partial y_k^t} \quad (3)$$

Here, $P(\mathbf{l}|\mathbf{x})$ in (3) can be efficiently calculated by forward-backward algorithm [18]

$$P(\mathbf{l}|\mathbf{x}) = \sum_{j=1}^{|\mathbf{l}'|} \alpha_t(j)\beta_t(j) \quad (4)$$

where $\mathbf{l}'$ is the modified label sequence for phone sequence $\mathbf{l}$ by adding `blank` to the beginning and end of $\mathbf{l}$ as well as the gap between every pair of neighbouring labels. Hence, the length of $\mathbf{l}'$ is larger than $\mathbf{l}$. j is used to denote the length of a modified label sequence. $\alpha_t(j)$ and $\beta_t(j)$ are the forward and backward probabilities at time t with length j. With equation (4), back propagation can be used to derive gradients of the parameters of the neural network (in this paper, uni-directional LSTM). Details of CTC training can be found in [18].

CTC model optimizes the conditional probability of label sequence by summing over all probabilities of the mapped paths, which include repeated labels and span of `blank` between pairs of labels. Therefore, the acoustic information output from CTC-trained model is concentrated (by learning the many-to-one function of $\mathcal{B}$) and the output distribution of CTC-trained model is peaky.

III. Phone Synchronous Decoding and CTC Lattice Generation

A. From Frame Synchronization to Phone Synchronization

In LVCSR, decoding is to find the best word sequence. By applying dictionary and language model to transform word sequence to CTC label sequence, the decoding formula in CTC-trained model is derived as

$$\begin{aligned}\mathbf{w}^* &= \underset{\mathbf{w}}{\operatorname{argmax}}\{P(\mathbf{w})p(\mathbf{x}|\mathbf{w})\} = \underset{\mathbf{w}}{\operatorname{argmax}}\{P(\mathbf{w})p(\mathbf{x}|\mathbf{l}_\mathbf{w})\} \\ &= \underset{\mathbf{w}}{\operatorname{argmax}}\left\{\frac{P(\mathbf{l}_\mathbf{w}|\mathbf{x})P(\mathbf{w})}{P(\mathbf{l}_\mathbf{w})}\right\} \quad (5) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}}\left\{P(\mathbf{w})\max_{\mathbf{l}_\mathbf{w}}\frac{P(\mathbf{l}_\mathbf{w}|\mathbf{x})}{P(\mathbf{l}_\mathbf{w})}\right\} \quad (6)\end{aligned}$$

Here, mono-phone CTC is taken as an example (the CTC label set consists of phone labels and the `blank` symbol). $\mathbf{w}$ is a word sequence and $\mathbf{w}^*$ is the best word sequence. $\mathbf{l}_\mathbf{w}$ denotes the phone sequence corresponding to $\mathbf{w}$[1]. $P(\mathbf{l}_\mathbf{w})$ in (5) is the prior probability of phone sequence.

For a certain CTC label sequence, forward probability is defined and approximated as (7), [18]

$$\begin{aligned}P(\mathbf{l}|\mathbf{x}) &= \sum_{\pi:\pi\in L',\mathcal{B}(\pi_{1:T})=\mathbf{l}}\prod_{t=1}^{T} y^t_{\pi_t} \\ &\cong \max_{\pi:\pi\in L',\mathcal{B}(\pi_{1:T})=\mathbf{l}}\prod_{t=1}^{T} y^t_{\pi_t} \quad (7)\end{aligned}$$

Therefore, (6) can be transformed to *frame synchronous Viterbi beam search* as below. Similar to the form in DNN-HMM, here network is traversed at each frame in an overall optimized search space [10].

$$\mathbf{w}^* \cong \underset{\mathbf{w}}{\operatorname{argmax}}\left\{P(\mathbf{w})\max_{\pi:\pi\in L',\mathcal{B}(\pi_{1:T})=\mathbf{l}_\mathbf{w}}\frac{1}{P(\mathbf{l}_\mathbf{w})}\prod_{t=1}^{T} y^t_{\pi_t}\right\} \quad (8)$$

Equation (8) reveals that $\pi_t = \texttt{blank}$ doesn't change the mapping result of $\mathcal{B}(\pi_{1:T})$. Hence, linguistic score keeps unchanged when $\pi_t = \texttt{blank}$. As for acoustic score, if all competing paths share the same span of `blank` frames, ignoring the scores of these frames does not affect the acoustic score rank (with softmax layer in CTC model, the `blank` acoustic score is approaching a constant of 1). Therefore, when speech waveform is in the long span of `blank`, any calculation of acoustic and linguistic scores becomes redundant because the linguistic search space does not change at all during that span. After the `blank` span, linguistic search space starts to change with the acoustic information from the next phonemic span. At that point, search should be continued. We name such characteristics as the *discontinuous linguistic search space* (DLSS) phenomenon of CTC-trained model. The DLSS phenomenon in CTC-trained model can be used to remove tremendous search redundancy due to `blank` frames.

Here, the set of *common* `blank` frames are defined as

$$U = \{u : y^u_{\texttt{blank}} \simeq 1\} \quad (9)$$

and equation (8) can be rewritten as

$$\begin{aligned}\mathbf{w}^* \cong \underset{\mathbf{w}}{\operatorname{argmax}}\Bigg\{&P(\mathbf{w})\max_{\pi:\pi\in L',\mathcal{B}(\pi_{1:T})=\mathbf{l}_\mathbf{w}}\frac{1}{P(\mathbf{l}_\mathbf{w})} \\ &\times\left\{\prod_{t\notin U} y^t_{\pi_t}\cdot\prod_{t\in U} y^t_{\texttt{blank}}\right\}\Bigg\} \quad (10) \\ = \underset{\mathbf{w}}{\operatorname{argmax}}&\left\{P(\mathbf{w})\max_{\pi:\pi\in L',\mathcal{B}(\pi_{1:T})=\mathbf{l}_\mathbf{w}}\frac{1}{P(\mathbf{l}_\mathbf{w})}\prod_{t\notin U} y^t_{\pi_t}\right\} \\ = \underset{\mathbf{w}}{\operatorname{argmax}}&\left\{P(\mathbf{w})\max_{\pi':\pi'\in L,\mathcal{B}(\pi'_{1:J})=\mathbf{l}_\mathbf{w}}\frac{1}{P(\mathbf{l}_\mathbf{w})}\prod_{j=1}^{J} y^{t_j}_{\pi'_j}\right\} \quad (11)\end{aligned}$$

where j is the index of the *phone sequence* (i.e. non-`blank` CTC label sequence), $\pi'_{1:J}$ is the phone-wise decoding path from phone 1 to J. J is the number of output phone labels ($J = T - |U|$) Comparing (8) to (11), the number of search iterations in (8) is T, while in (11) it is J. J is usually significantly smaller than T. Therefore, based on the DLSS phenomenon, frame synchronous Viterbi beam search has been transformed to *phone synchronous Viterbi beam search*. The main differences between frame synchronous decoding (FSD) and phone synchronous decoding (PSD) include

1) *Different Information Rate:* In FSD, both acoustic and linguistic information are processed at each frame, forcing both information rates to be the same as the frame rate of acoustic feature. In contrast, in PSD, acoustic information is processed by frame rate of acoustic feature, while linguistic information is processed phone by phone (phonemic rate of acoustic model inference). The different rates of processing in acoustic and linguistic information remove tremendous search redundancy.
2) *Adjustable Search Interval:* In FSD, WFST network is traversed at fixed equal interval (even *multi-frame deep neural networks decoding* [4] traverses linguistic search space by longer but still equal interval). In contrast, in PSD, the search interval is self-adjusted (smart and without performance deterioration) to remove search redundancy due to `blank` frames, which brings about significant efficiency improvement in decoding.

It is worth noting that within the FSD framework, approaches to adjust search interval have also been investigated, such as *frame skipping* (FS) [4], [5], or more sophisticated *variable frame rate* schemes (VFR) [6], [7]. Although these methods all achieve frame skipping effect and consequently improve decoding speed, they only focus on acoustic features and do not consider linguistic search space. As previously discussed, different information rates of knowledge sources inevitably lead to time granularity difference during search. Simply replying on feature level analysis may not capture and address the difference

[1]Here, only single pronunciation dictionary is considered for notation simplicity.

accurately. Hence, the search interval adjustment, or variable frame rate, schemes within the FSD framework has limitation in optimising search efficiency. In contrast, within the proposed PSD framework, frame rate is adjusted and adapted automatically to linguistic space change. PSD not only removes heuristics in frame rate analysis, but also achieves search efficiency and less acoustic information loss.

Alternatively, search space compression is another direction to fulfill the peaky property of CTC trained model during decoding. A well engineering method is to adopt traditional adaptive beam pruning strategy [26] by applying distinct pruning strategies in `blank` frames and phonemic frames. However, such method is hard to achieve the best efficiency, and lack of theoretical support and carefully research. Essentially, adaptive beam pruning strategy achieves speed-up in both acoustic candidate selection and search space compression, while PSD directly speeds up by longer decoding steps.

Recently, novel HMM topology was proposed in LF-MMI [27], which holds a similar many-to-one mapping as $\mathcal{B}$ function discussed in Section II, except the difference that each triphone gets its own version of the `blank`. In our experiment, the output distribution of LF-MMI is not so peaky like that in CTC. However, HMMs in LF-MMI emit phone units with extra `blank`, as well. Therefore, it is interesting to apply PSD framework in LF-MMI trained model after some specific treatment in `blank`, and may be the future work.

B. Search Space Analysis

As indicated in Section III-A, In phone synchronous decoding (PSD), there is almost no search loss compared to standard frame synchronous decoding (FSD). On the other hand, compared to FSD, the rate of linguistic information processing is greatly reduced, because of the removal of large number of `blank` frames in the decoding paths of CTC-trained model. It is therefore interesting to analyze the change of search space and decoding errors. Two metrics are defined for this purpose.

Network Traversal Reduction Rate, λ, is defined as the average of `blank` frame percentages of test utterances

$$\lambda = \frac{1}{N}\sum_{n=1}^{N}\frac{\#\{U^{(n)}\}}{T^{(n)}} \tag{12}$$

where n is the index of test utterance and N is the total number of utterances, $\#\{U^{(n)}\}$ and $T^{(n)}$ are the number of `blank` frames and the total number of frames of utterance n. By definition, λ denotes the percentage of linguistic search space traversal reduction in PSD for a given set of test utterances.

Even in non-`blank` frames, not all phones have noticeable posteriors from the CTC model. Hence, λ is just a lower bound of the actual search space compression in practice. To get a more realistic estimation, we further define an additional term, *Active Phone Rate*, β. It is calculated as the percentage of active phones (phones with posteriors larger than a small threshold) with respect to all phones for a given set of test utterances. With λ and β, the overall *Theoretical Compression Rate*, R, is defined as below

$$R = 1 - (1-\lambda)\times\beta \tag{13}$$

By definition, R is the overall measure of the search space compression yielded by PSD. In practice, λ is usually about 0.8, and β is around 0.1, yielding a very effective theoretical compression rate $R = 0.98$. More experimental analysis is given in Section V-A. Thanks to the tremendous reduction in active tokens, a wider beam can be used to achieve better performance. That's another superiority of PSD to FSD.

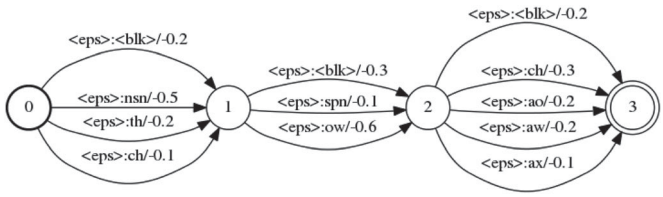


Fig. 1. CTC Lattice Example.

TABLE I
ACOUSTIC INFORMATION OF CTC LATTICE EXAMPLE

Time	phone label : acoustic score
0.4 s	$< blk >: 0.2 \quad nsn : 0.5 \quad th : 0.2 \quad ch : 0.1$
0.9 s	$< blk >: 0.3 \quad ow : 0.6 \quad spn : 0.1$
1.5 s	$< blk >: 0.2 \quad ch : 0.3 \quad ao : 0.2 \quad aw : 0.2 \quad ax : 0.1$

C. CTC Lattice - Extremely Compact Acoustic Information Preserver

Given the above discussion, *CTC lattice* is proposed, as a preserver of phone level acoustic information. CTC lattice can fulfil the search optimization advantage in Section III-B, while providing a compact and flexible form for further utilization.

Figure 1 is an example of CTC lattice in WFST form. Suppose phone level acoustic scores from CTC model is as table I, WFST can be built with "sausage" style. Here each span between two phonemic frames is connected by arcs with each phone label as WFST output label and its negative acoustic score as arc weight. The input label is set to *epsilon*, $<eps>$, so that further WFST optimization can be done.

From table I, phone information are only stored at 3 time frames. Compared to frame level phone posteriors, CTC lattice is extremely compact in acoustic information compression. With further post-processing, consecutive phones and `blank` can also be removed, which is discussed in Section IV-A. Despite the compactness, almost all acoustic information is well preserved in CTC lattice due to the peaky posterior property of CTC model. The compactness and effectiveness of acoustic information preservation leads to wide applications as discussed in the below section.

IV. SPEECH RECOGNITION WITH CTC LATTICE

In ASR, there are mainly two kinds of tasks, large vocabulary continues speech recognition (LVCSR) and keyword spotting

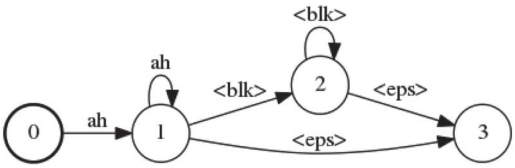


Fig. 2. Example of Dynamic Expanding Acceptor in Acoustic Search Space.

(KWS) . LVCSR aims to recognize naturally spoken utterances with large vocabulary of words. KWS aims to accurately and efficiently detect a small number of predefined keywords in continuous speech. Both of LVCSR and KWS tasks can be implemented in either *top-down integrated* decoding architecture [28], [9], [29], or *bottom-up modular* decoding architecture [30], [16]. This section will discuss the technical details of applying PSD derived CTC lattice to both tasks within the two different decoding architectures.

A. Top-Down Integrated Search

In the architecture, WFST [8] is proposed to offline combine different knowledge sources and perform search space optimization in advance. Then Viterbi beam search is applied in the overall search space to get the final result.

1) Search Space Construction: Search space of phone synchronous decoding can be constructed into two parts similar to [31], referred to as *acoustic search space* and *linguistic search space* respectively.

In acoustic search space, a dynamic expanding acceptor is constructed for each model unit in CTC, to filter out `blank` labels and repeating phone labels. Figure 2 is an example of such structure for a certain model unit(*ah*as an example). The dynamic expanding acceptor is applied on-the-fly during acoustic space search.

After going through acoustic space, token passing algorithm [32] is performed on a pre-compiled static phonemic weighted graph (linguistic search space). Linguistic search space is constructed and optimized by WFST operation before decoding. In this paper, the model unit of CTC is mono-phone[2]. Therefore, rather than using $HCLG$ as in [9] (H for acoustic model and transition model, C for phone context dependency model, L for pronunciation lexicon and G for language model), only LG is used here. In LVCSR task, G is grammar [33] or N-gram [34] language model.

Compared to the standard static WFST based search graph for context dependent HMM model in [9], the search graph of phone synchronous decoding is reduced by approximately 5 times because of removing HC graph. Compared to the TLG graph proposed in [23] (T for token mapping WFST), there's also 2-3 times of size reduction by removing T graph.

2) Search Algorithm: Phone synchronous decoding algorithm is summarized as Algorithm 1.

[2]In tri-phone based CTC [20], it is also possible to use PSD. In this case, acoustic search space is designed with similar acceptor structure in Figure 2, while CLG is used as linguistic search space.

Algorithm 1: Phone Synchronous Viterbi Beam Search(S, E, Q, T).

1: $Q \leftarrow S$ ▷ initialization with start node
2: **for** each $t \in [1, T]$ **do** ▷ frame-wise NN Propagation
3: $F \leftarrow NNPropagate(t)$
4: **if** !isBlankFrame(F) **then** ▷ phone-wise WFST search
5: $Q \leftarrow ViterbiBeamSearch(F, Q)$
6: **end if**
7: **end for**
8: $\hat{B} \leftarrow finalTransition(E, S, Q)$ ▷ to reach end node
9: backtrace($\hat{B}$)

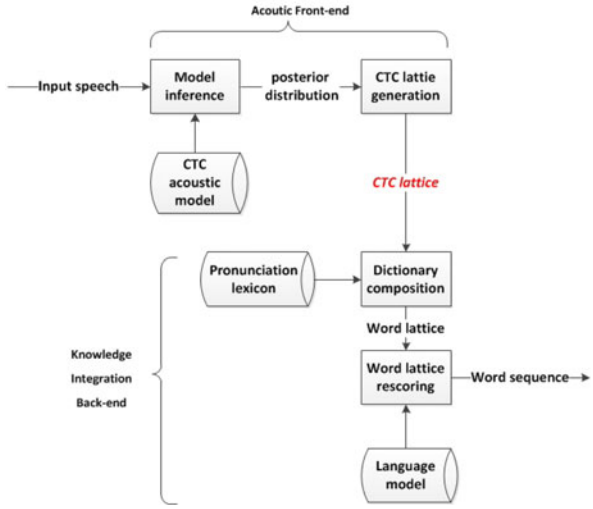


Fig. 3. WFST-based Modular Composition with CTC Lattice.

Here, S and E are start and end node of the precompiled WFST network. Q preserves active tokens, $\hat{B}$ preserves decoding paths. T is the total number of frames. $NNPropagate(t)$ is to do acoustic model inference at each frame. $isBlankFrame(F)$ is to detect whether a frame is `blank` or not. $ViterbiBeamSearch(F, Q)$ is the standard Viterbi beam search algorithm in FSD but only conducted at phone level in PSD. $finalTransition(E, S, Q)$ is to traverse to the end node of WFST as in [35].

B. Bottom-up Modular Search Method

In bottom-up modular search, phone lattice is first generated from acoustic model inference as front-end. Then other knowledge sources are applied to the lattice to get final results.

1) WFST-based Modular Composition: The architecture is shown in Figure 3. The first step is to transform phone level CTC lattice to word level lattice using WFST composition with a dictioinary. The pipeline of this procedure is

$$W \leftarrow epsremoval(P \circ T \circ L) \tag{14}$$

where P is CTC lattice and L is lexicon WFST constructed from a dictionary. T is designed to filter out `blank` symbol and repeating phone label as in [23] (T stands for token mapping

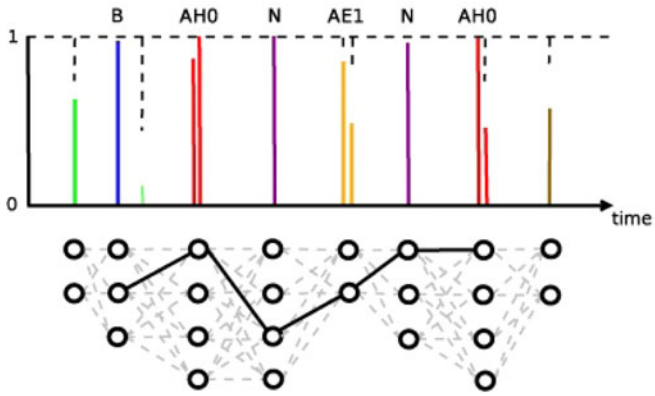


Fig. 4. CTC output of a speech segment and the corresponding CTC lattice. Different colours represent the posteriors of different phones. The black line in the lattice indicates a potential path and the grey dashed lines are all valid connections.

WFST). Because there are many *epsilon* labels in P, *epsilon removal* [9] is performed on the composition result. Finally, P is transformed to word lattice W and ready for applying language model. It is worth noting that, with CTC lattice, it is easy to employ different dictionaries of the same phone set during decoding. Once phone level lattice is transformed to word level lattice, it's trivial to apply different types of language models, which is usually referred to as lattice rescoring. For example, high order n-gram [35] or RNN [36], [37] can be used.

Since large portions of `blank` frames are removed during phone synchronous decoding, given the same lattice oracle error rate, CTC lattice is much more compact than phone lattice generated from frame synchronous decoding. This is because there are less arcs with the same phone label but slightly different time boundaries. Therefore, it is more convenient to perform rescoring using complex language model on CTC lattice.

2) MED-based Phone Lattice Pattern Matching: Based on CTC lattice, a *minimum edit distance* (MED) based phone lattice pattern matching approach for keyword spotting task is proposed as shown in figure 4.

Suppose $\mathbf{l_w}$ is the phone sequence of predefined keyword and $\mathbf{l}$ be a hypothesized phone sequence. For a speech utterance, let L_H represent the corresponding CTC phone lattice containing all possible phone sequences. Then, the probability of target keyword $\mathbf{l_w}$ existing in L_H is

$$P(\mathbf{l_w}|L_H) \propto P(L_H|\mathbf{l_w})P(\mathbf{l_w}) \approx P(\mathbf{l_{max}}|\mathbf{l_w})P(\mathbf{l_w}) \quad (15)$$

where $\mathbf{l_{max}}$ is the most likely phone sequence in L_H given $\mathbf{l_w}$. $P(\mathbf{l_w})$ is the prior probability of the phone sequence of keyword $\mathbf{w}$. Since $\mathbf{l_w}$ is given, whether the keyword $\mathbf{l_w}$ exists or not in the test utterance is dependent on the value of $P(\mathbf{l_{max}}|\mathbf{l_w})$. Here, $\mathbf{l_{max}}$ is defined as

$$\begin{aligned}\mathbf{l_{max}} &= \underset{\mathbf{l}}{\operatorname{argmax}} P(\mathbf{l}|\mathbf{l_w}) = \underset{\mathbf{l}}{\operatorname{argmax}} \frac{P(\mathbf{l_w}|\mathbf{l})P(\mathbf{l})}{P(\mathbf{l_w})} \\ &= \underset{\mathbf{l}}{\operatorname{argmax}} P(\mathbf{l_w}|\mathbf{l})P(\mathbf{l}) \end{aligned} \quad (16)$$

where $P(\mathbf{l})$ is the CTC probability of $\mathbf{l}$, which can be calculated using equation (7).

$P(\mathbf{l_w}|\mathbf{l})$ is estimated through multiplying the probabilities of each edit operation when computing the minimum edit distance (MED) between $\mathbf{l}$ and $\mathbf{l_w}$,

$$P(\mathbf{l_w}|\mathbf{l}) \triangleq \prod_{i=1}^{MED(\mathbf{l_w},\mathbf{l})} P(op_i|\mathbf{l_r} = \mathbf{l_w}, \mathbf{l_e} = \mathbf{l}) \quad (17)$$

$$P(op_i|\mathbf{l_r} = \mathbf{l_w}, \mathbf{l_e} = \mathbf{l}) = \begin{cases} P(ins(e_i)) & \text{if } op_i \in I \\ P(del(r_i)) & \text{if } op_i \in D \\ P(r_i|e_i) & \text{if } op_i \in S \end{cases} \quad (18)$$

where $MED(\mathbf{l_w}, \mathbf{l})$ indicates the minimum number of edit operations (insertion, deletion and substitution) between $\mathbf{l_w}$ and $\mathbf{l}$. $P(op_i|\mathbf{l_r} = \mathbf{l_w}, \mathbf{l_e} = \mathbf{l})$ denotes the probability of the i-th edit operation op_i between $\mathbf{l_e}$ and $\mathbf{l_r}$, given that the reference sequence $\mathbf{l_r}$ is $\mathbf{l_w}$ and the hypothesis sequence $\mathbf{l_e}$ is $\mathbf{l}$. Since MED algorithm is deterministic and both $\mathbf{l_e}$ and $\mathbf{l_r}$ are given, each edit operation is known.

According to the type of op_i, there are three kinds of probabilities: the probability of insertion operation $P(insert(e_i))$, deletion operation $P(delete(r_i))$ and substitution operation $P(r_i|e_i)$. I, D and S denote the sets of insertion, deletion and substitution operations respectively. e_i and r_i are the phones involved in op_i (insertion and deletion involve one phone and substitution involves two) and they belong to hypothesis $\mathbf{l_e}$ and reference $\mathbf{l_r}$ respectively. These three kinds of probabilities can be estimated from prior knowledge. In this paper, they are estimated directly through a phone-level confusion matrix [38] calculated by comparing the phone alignment between an ASR hypothesis and the corresponding reference. Instead of enumerating all possible $\mathbf{l}$ in the lattice, (16) can be efficiently computed in $O(Nnd)$ time using dynamic programming, where N is the total number of nodes in the lattice, and d is the depth of the lattice [39]. Normally, n and d are small ($n < 15$ and $d < 5$), and N depends on the number of phones of a utterance. Therefore, the hypothesis search is efficient.

Empirically, for different phones, CTC model may have different modelling effects. Therefore, the threshold of a keyword should be dependent on its phone sequence. Prior statistics can be used to estimate an optimal contribution factor for each phone on a valid set. Besides, a special label *wb* is introduced to model the word boundaries in CTC model [40]. The motivations are two folds: (a) increasing the length of short keywords to avoid its phone sequence being a substring of a longer word, (b) *wb* helps the search algorithm to stride across those misidentified spikes.

V. Experiment

In this section, the quality of CTC lattice is analyzed first. Then, experiments of LVCSR on an English switchboard task and KWS on a Wall Street Journal task are reported in detail.

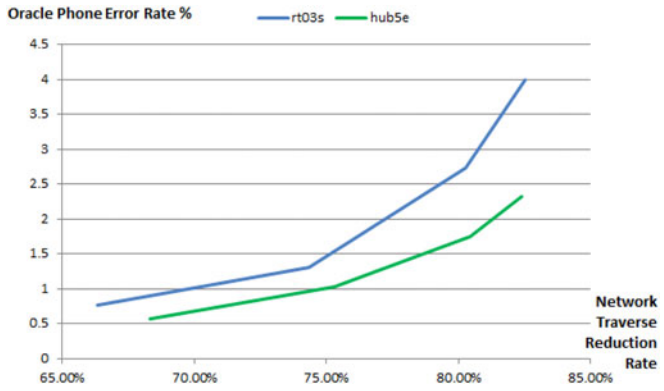


Fig. 5. Network Traversal Reduction λ v.s. Oracle Phone Error Rate.

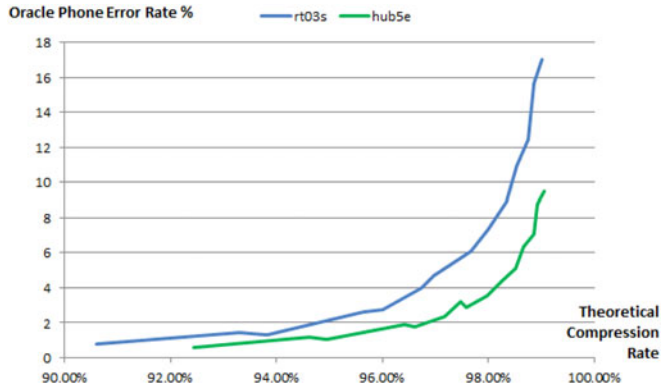


Fig. 6. Theoretical Compression Rate R v.s. Oracle Phone Error Rate.

A. CTC Lattice Quality Analysis

As shown from equation (10) to (11) in section IV-B, CTC lattice is an extremely high quality phone lattice. Experimental analysis to demonstrate this is reported in this section. In the experiments, the LSTM-CTC acoustic model was trained on a 310 hours English Switchboard task (details of setup are listed in V-B1) and tested on the NIST 2000 Hub5e set (referred to as hub5e) and Rich Transcription 2003 set (referred to as rt03 s).

Oracle phone error rate (OPER) is used as the measurement of CTC phone lattice quality. It directly reflects the overall quality of phone lattice and is calculated as the error rate of the best possible phone sequence existing in the lattice w.r.t. the reference phone sequence [41].

As discussed in section III-B, network traversal reduction rate λ reflects the linguistic search space reduction caused by removing all `blank` frames. Since whether a frame is `blank` is determined by the probability of `blank` symbol as shown in equation (9), different thresholds may result in different λ and consequently different OPER. To investigate the relationship between linguistic search space reduction using PSD and the quality of the resultant CTC lattice, we plotted OPER versus network traversal reduction rate λ, as shown in figure 5. It can be observed that even after 80% of frames are regarded as `blank` frames and removed, the OPER is still less than 3%. This will lead to significant decoding speed-up with little information loss. Given the research of the relationship between phone error rate (PER) and word error rate (WER) in [15], such rate of OPER is sufficiently low for further combination with other knowledge sources.

λ only shows linguistic search space reduction, overall search space reduction should also take into account the rate of actual active phones. As indicated in section III-B, the theoretical compression rate R reflects overall search space reduction as well as the compactness of CTC lattice. Figure 6 shows the relationship between theoretical compression rate R and OPER. It can be seen that even if 95% overall search space is compressed, the corresponding OPER is still very low (less than 3%).

In light of the previous experiments, we can choose appropriate `blank` frame threshold (corresponding to λ) and CTC posterior prune threshold (corresponding to β) to get a good trade-off between search space reduction and OPER increase.

TABLE II
STATISTICS OF SEARCH SPACE REDUCTION AND OPER

Method	Testset	$\lambda(\%)$	$\beta(\%)$	$R(\%)$	$OPER(\%)$
CTC	hub5e	75	10	97.5	3.22
Lattice	rt03s	74	11	97.0	4.74
Alignment	hub5e	77	5	98.8	0
	rt03s	75	7	98.2	

Table II shows the chosen λ and β for the rest experiments in the paper[3]. λ is directly related to decoding speed, while β is also related to compactness and quality of CTC lattice in preserving acoustic search space. The resultant theoretical compression rate R and OPER are also given. It can be observed that the two indicators show impressive search space compression with very little potential information loss.

The first row block of table II shows the statistics of PSD and CTC lattice in actual decoding. To further investigate how good the chosen parameters are, ideal statistics (best possible search space reduction and OPER) are also calculated. To get the ideal statistics, forced-alignment [33] was performed on CTC paths (the method is the same to alignment step in sequence training of CTC model [20]) and accurate phone or `blank` labels were gotten for each frame. It is then trivial to calculate an ideal λ, i.e. the actual `blank` frame rate. Given the aligned phone label for each frame, it is possible to find the aligned phone with lowest CTC model posterior and set the global β accordingly, to allow all aligned phones exist in the resultant CTC lattice. Obviously, λ and β chosen in this way will lead to zero OPER. It can be observed that the ideal search space reduction R (without any information loss) is very similar to the R obtained from the actual PSD process. This demonstrates the efficiency of PSD and CTC lattice.

B. LVCSR task

1) Experimental Setup: The experiments were conducted on an English Switchboard task. A subset of 51 hours data from 310

[3]`blank` frame threshold and CTC posterior prune threshold are fixed in the whole paper and mainly related to the peaky degree of the acoustic model, essentially. These parameters are not sensitive to testset, which is discussed after Table IV, as well.

TABLE III
WER COMPARISON BETWEEN CI-CTC, CD-CTC & CD-HMM

			WER (%)			
			hub5e		rt03s	
Dataset	Context Dependency	Model	swb	callhm	fsh	swb
50 hr	CD	DNN-HMM	19.8	33.2	24.5	34.6
	CD	LSTM-HMM	19.5	32.4	24.1	32.8
	CI	LSTM-CTC	29.3	45.0	35.6	45.4
310 hr	CD	DNN-HMM	16.7	29.3	20.9	30.7
	CD	LSTM-HMM	14.9	26.9	19.4	28.3
	CI	LSTM-CTC	18.7	33.3	24.3	34.5
	CD	LSTM-CTC	18.0	31.1	23.7	32.8

hours Switchboard dataset was randomly chosen to form a small training set for development. Both context-dependent (CD) and context-independent (CI) LSTM-CTC models were built. The training procedure and configuration are similar as [24]. 7-layer CD-DNN-HMM and 3-layer CD-LSTM-HMM were built as baselines. CD-DNN-HMM and CD-LSTM-HMM models used 8775 tri-phone states as output, while CD-LSTM-CTC model used 1326 tri-phones and CI-LSTM-CTC model used 46 phones as output. Filter bank coefficients were used as input features for all models. All LSTM models employed projection technique [42]. All models were designed with around 2.5-3 M parameters to get fair comparison (except that CI-CTC model has significantly less parameters than the others). An interpolated trigram language model trained on 2000 hours of Fisher transcription was used for decoding.

The decoder used in all experiments is an internal optimised WFST decoder. During testing, the decoding machine setup is *Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00 GHz*. The NIST 2000 Hub5e set (referred to as hub5e) and Rich Transcription 2003 set(referred to as rt03 s) are used as test sets. Full tests were evaluated for baseline comparison and only the switchboard parts were used in latter experiments.

2) Baseline Performance: Performance of LSTM-CTC models are compared with state-of-the-art DNN and LSTM models. Both context-independent (CI) and context dependent (CD) LSTM-CTC were built.

From table III, the performance is compatible with [23], [24], [27], e.g., in [27], the attempts to get CTC to beat cross-entropy trained systems on mere hundreds of hours of data were also unsuccessful. We suspect the reason is that the `blank` in CTC is hard to achieve good modelling effect with limited data. Correspondingly, it can be observed that CTC model performs better with more data. Although in 310-hours dataset, there is still some gap between LSTM-HMM and LSTM-CTC models, the gap becomes narrower than the 50-hours setup. Previous works [25], [22] showed that with larger training dataset, CI-phone-CTC model performs better than CD-state-HMM model. Our previous work [43] also shares similar finding in large dataset. Since the focus of this paper is on search algorithm, latter experiments are still conducted on CI-LSTM-CTC trained on 310-hours of data for better research comparison. It is noted that the conclusion of this paper is also valid on CD-LSTM-CTC model.

TABLE IV
PHONE SYNCHRONIZATION COMPARED WITH FRAME SYNCHRONIZATION

Testset	Sync	WER (%)	S-RTF	RTF	Active Tokens
hub5e	Phone	18.8	0.022 (**3.4X**)	0.056	513(−**77%**)
-swb	Frame	18.7	0.075	0.11	2221
rt03s-swb	Phone	34.4	0.022(**3.3X**)	0.055	511 (−**77%**)
-swb	Frame	34.5	0.073	0.12	2211

3) Top-Down Integrated Phone Synchronous Decoding:

1) *Speed-up of Phone Synchronous Decoding (PSD)*

Experiment results on phone synchronous decoding (PSD) are listed in Table IV, with the most popular WFST-based CTC implementation (FSD) [23] as the baseline.

Real time factor (RTF), the percentage of decoding time w.r.t. wave time, is a common measurement for decoding speed. Result shows that, there's no ASR performance degradation in PSD, while achieving **2-3***X* overall speed-up compared with the standard FSD CTC implementation [23]. Decoding time includes both neural network propagation time and WFST search time. Since PSD mainly speeds up the latter one, pure WFST search time divided by waveform length is individually listed as *S-RTF* in the table, which reveals **3-4***X* speed-up in it. Besides, the decrease of active tokens in table IV also parallel reflects both *S-RTF* here and λ in Table II.

Besides, the speed-up rates in hub5e-swb and rt03s-swb are similar. As discussion in Section V-A, the fixed parameters related to β and λ is only related to the acoustic model but not sensitive to different testsets. Our preliminary trials on noisy testset also shows similar speed-up with slight decrease (relative 10%), but without loss in ASR performance. Research on the decrease in speed-up can be the future work.

Our other PSD experiments also show similar 3-4X speed-up with CD-phone-CTC model with no loss in ASR performance. The reason is that the outputs of acoustic model are still phone units with a single extra `blank`. Although there are more phone units, the single `blank` make the model peaky enough to distinguish `blank` frames and phonemic frames.

1) *Speed-up Robustness to LM Size*

In the experiment, language model (LM) size was increased to test the robustness of speed-up (extendibility to more complex linguistic search space) from frame to phone synchronous decoding. N-gram LM was chosen here, but the results can be easily extended to other LM, such as RNN [44]. In WFST-based decoder, RTF shows nearly linear relationship with average number of *active token* [35], [10], so active token number is used to measure the speed of decoding.

Figure 7 shows that active token number in PSD is almost unchanged with the growing size of language model. In the same experiment, active token number of CD-state-NN-HMM system is always far more than PSD, especially when LM scales up. Besides, active token number in FSD is also growing distinctly faster compared with PSD. It is then concluded that the speed-up achieved by PSD is robust to the increase of LM search space.

1) *Comparison with Other Frame Rate Changing Methods*

As previously discussed, PSD can be regarded as a variable frame rate approach. Hence, it is interesting to compare it

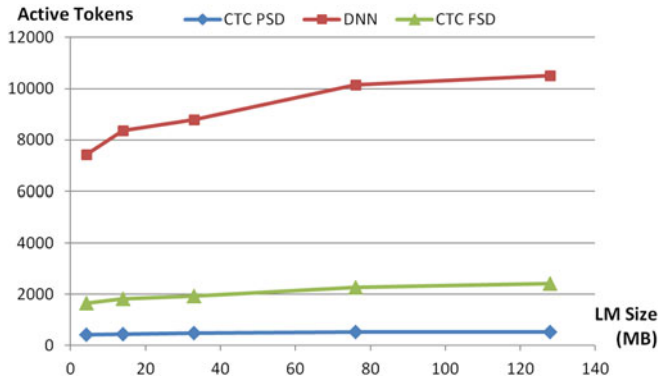


Fig. 7. LM scale-up v.s. active tokens in phone/frame synchronous decoding. For clarity, only `hub5e-swb` is plotted and the rest sets have similar trends.

TABLE V
COMPARISON BETWEEN PSD AND OTHER FRAME RATE CHANGING METHODS

Testset	Method	WER (%)	S-RTF	RTF
hub5e-swb	PSD	18.8	0.022	0.056
	VFR	24.7	0.038	0.058
	FS	18.5	0.039	0.057
	PSD + FS	18.4	0.017(**4.4X**)	0.035
rt03s-swb	PSD	34.4	0.022	0.055
	VFR	39.5	0.038	0.057
	FS	34.1	0.039	0.056
	PSD + FS	34.2	0.017(**4.3X**)	0.036

with various frame rate changing approaches within the FSD framework. Here, *frame skipping* (FS) and *variable frame rate* (VFR) schemes were investigated. With LSTM-CTC model, FS was implemented similar to [5] but without procedure of posterior copying because such procedure mainly deals with HMM state level loops and transitions, which brings about extra burdens in WFST search. VFR was implemented in a typical setup as in [45]. Both *S-RTF* and *RTF* were evaluated because FS and VFR affect both indicators while PSD mainly affects the latter one.

Comparing table V to table IV, FS applied to CTC model can speed up both *S-RTF* and *RTF* by 2 times without performance deterioration. This is in accordance with the observation in [5] and similar to the results in DNN-HMM [4] and LSTM-HMM [5]. Meanwhile, VFR leads to significant degradation with speed-up rate (**2X**).

Compared with PSD, FS shows competitive efficiency. The reason is that FS speeds up both neural network propagation and WFST search[4], while PSD mainly deals with the latter one (and it speeds up more in the latter one). However, PSD can be further combined with FS. In this way, *PSD+FS* in Table V shows much better efficiency compared with both PSD and FS (accumulatively **4-5X** speedup on FSD). It is inspiring that feature level and model level frame rate changing methods can be combined together to achieve better gains.

[4]It is worth noting that, in [4] and [5], the traditional *frame skipping and posterior copying* can only speed up neural network propagation.

TABLE VI
WER PERFORMANCE OF DIFFERENT PSD DECODING METHODS WITH CTC LATTICE

	PSD with CTC Lattice	
Testset	Integrated	Modular
hub5e-swb	18.7	18.5
rt03s-swb	34.5	34.1

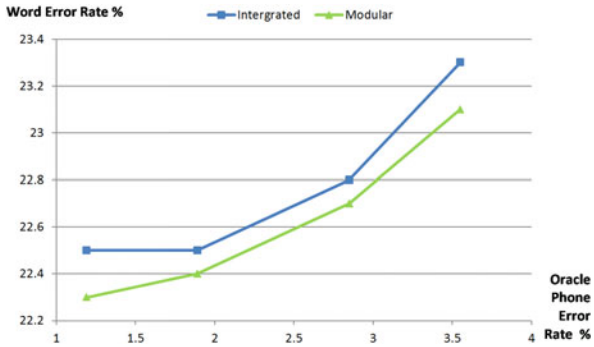


Fig. 8. Phonemic Error Robustness of Different Decoding Methods. For clarity, only `hub5e-swb` is plotted and the rest sets have similar trends.

4) Bottom-up Modular Search with CTC Lattice: Bottom-up modular search with CTC lattice was implemented based on WFST composition as described in section IV-B1.

Results in Table VI show consistent WER improvement compared with the integrated PSD search in the previous section. The reasons are mainly two folds. Firstly, there is no search error in modular method, while beam search is the main source of search error in integrated method. Secondly, as indicated in III-B, CTC lattice is compact and precise, which is the main superiority to prior works [14], [46], [30], [17]. However, since there is no other knowledge source introduced except for acoustic and language model, the improvement is mild.

To further investigate the relationship between final WER and OPER of CTC lattice, integrated and modular search were performed on CTC lattice with different qualities. The comparison on the hub5e-swb test set is shown in figure 8.

It can be observed that modular search is consistently better than integrated search on different OPERs. Also, WER of modular search has less fluctuation w.r.t. OPER. The reason is that modular search is performed on CTC lattice generated without search error in acoustic space, while integrated beam search introduces random search errors during CTC lattice generation. Since the final performance of modular search is more reliably related to OPER of CTC lattice, improving OPER is very likely to improve WER as well. Besides, modular search also keeps large improvement space for combination with richer knowledge sources (e.g., phonemic error model [15], prosody model [14]).

C. KWS task

A speaker-independent 5k vocabulary dataset of the Wall Street Journal (WSJ0) corpus [47] was used to evaluate the

TABLE VII
KEYWORD-FILLER HMMS VS. LSTM-CTC USING PSD WITH MED

Model	#Param	Search	EER	FOM	S-RTF	RTF
GMM-HMM	5.4M	FSD	6.4	71.8	—	—
DNN-HMM	2.0M		5.1	75.5	0.036	0.074
LSTM-CTC	813K	PSD+MED	3.5	85.2	0.005	0.031

proposed CTC lattice based KWS. Words or phrases which appear at least 5 times and whose length is between 3 and 12 phones were randomly selected as the keywords. In total, 50 keywords were used. The evaluation metrics are Equal Error Rate (EER) and Figure of Merit (FOM) [48]. For EER values, smaller is better. For FOM values, larger is better. Both EER and FOM are obtained by sweeping fixed thresholds.

24-dimensional log filter-bank coefficients with their first and second derivatives at 10 ms fixed frame rate was used to build the LSTM-CTC model. To accelerate KWS, LSTM Projected (LSTMP), was used [42]. The LSTM consists of 2 hidden layers with 384 memory blocks per layer, and the size of the linear projection layer is 128. The output label of LSTM are 69 phones from the CMU pronunciation dictionary plus `blank` symbol, and *wb* symbol to model word boundaries.

For performance comparison, conventional *keyword-filler* HMMs [49] were trained as baselines. The keyword models estimate the keyword likelihood given observation feature sequences, and the filler model represents all non-keyword speech. The keyword-filler HMM topology is identical to the one used in [50]. Through Viterbi decoding, if the best path passes a keyword model then the corresponding keyword is determined as detected. The EER and FOM were obtained by sweeping the transition probabilities between keyword and filler models. This transition probability can be regarded as the prior of keywords. Clustered cross-word tri-phone HMMs were used as baselines[5]. A GMM system with 40 Gaussian mixtures and a DNN system of 4 hidden layers with 512 nodes per layer were built. The acoustic feature for GMM is 13-dimensional cepstral mean normalized MFCC coefficients with their first and second order derivatives. DNN has an 11-frame context window with 5 extended frames on the left and right. Both HMM systems have 1689 clustered triphone states.

The performances of different KWS systems are shown in Table VII. Here, keyword-filler structure is compiled into WFST form and used for frame synchronous decoding with GMM-HMM and DNN-HMM systems. Word boundary is used in LSTM-CTC model [40] and PSD with MED based pattern matching approach is employed as described in section IV-B2. In calculation of *S-RTF* and *RTF*, we only consider single keyword detection speed. The results indicate that the proposed KWS approach using PSD and MED based pattern matching achieves significant gains over the traditional Keyword-Filler HMM approach (around 30% relative EER reduction, and 12% relative FOM increase), demonstrating the effectiveness of the proposed methods. Besides, the LSTM-CTC model also has much less parameters than the baselines.

It can also be observed that DNN-HMM system is 2.4 times slower than the LSTM-CTC system[6]. Firstly, this is because LSTM-CTC has 60% less parameters in its acoustic model. Secondly, the Keyword-Filler structure in [50] results in large search space especially for systems with clustered tri-phone states (context-dependent). In contrast, due to the use of PSD and MED-based phone lattice pattern matching method, the KWS searching speed of LSTM-CTC is 7.2 times faster. It is worth noting that with the current MED-based matching algorithm, decoding time grows linearly with the number of predefined keywords. Hence, if multiple keywords are to be detected simultaneously, the *PSD+MED* algorithm will be slower than the Keyword-Filler decoding approach. However, in common business applications (e.g., Google voice search [51]), the number of keywords is usually small and this increase is not likely to be a big issue.

VI. CONCLUSION

The output of CTC model is usually peaky and this results in large number of `blank` frames during decoding. These `blank` frames do not change the linguistic search space and search at these frames are therefore redundant. In this paper, by removing the `blank` frames from linguistic search space, traditional *frame synchronous decoding* (FSD) can be transformed into *phone synchronous decoding* (PSD). PSD can be viewed as a hybrid decoding framework of beam search and A* search because of its self-adjusted decoding interval to remove tremendous search redundancy due to `blank` frames from CTC-trained model. With PSD, compact and precise phone-level CTC lattice can be produced. Two modular search approaches based CTC lattice are proposed for LVCSR and KWS tasks respectively. Experiments show that PSD can achieve 3-4 times decoding speed-up compared to traditional FSD, with no performance deterioration. It can also be combined with feature level frame rate changing method to get further speed-up. With CTC lattice, in LVCSR task, modular search can achieve further WER improvement over integrated search and is more convenient to incorporate other knowledge sources. In KWS task, experiments demonstrated that MED based pattern matching on CTC lattice can significantly improve the KWS accuracy as well as the matching speed compared with the traditional keyword-filler structure. Future work will explore more usage of CTC lattice, such as integrating CTC lattice with more knowledge sources to fulfil the distinct OPER performance of it, e.g., phonemic error model [52], prosody model [53] and more complex language model [36], [37], [54], [55]. Besides, it is promising to apply PSD framework in HMM topology model, e.g., LF-MMI [27].

ACKNOWLEDGMENT

The author would like to thank the speech technology group of AISpeech Ltd. for infrastructure support and valuable technical discussions.

[5]Since the KWS task in this paper concerns unrestricted keyword vocabulary, keyword-specific approaches such as [50] are not appropriate baselines.

[6]Since GMM-HMM is slower than DNN-HMM due to larger model size, we did not include its RTF number.

REFERENCES

[1] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[2] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge, MA, USA: MIT Press, 1997.

[3] G. D. Forney, "The viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[4] V. Vanhoucke, M. Devin, and G. Heigold, "Multiframe deep neural networks for acoustic modeling," in *Proc. 2013 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2013, pp. 7582–7585.

[5] Y. Miao, J. Li, Y. Wang, S.-X. Zhang, and Y. Gong, "Simplifying long short-term memory acoustic models for fast training and decoding," in *Proc. 2016 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2016, pp. 2284–2288.

[6] K. Ponting and S. Peeling, "The use of variable frame rate analysis in speech recognition," *Comput. Speech Lang.*, vol. 5, no. 2, pp. 169–179, 1991.

[7] Z.-H. Tan and B. Lindberg, "Low-complexity variable frame rate analysis for speech recognition and voice activity detection," *IEEE J. Sel. Topics Signal Process.*, vol. 4, no. 5, pp. 798–807, Nov. 2010.

[8] M. Mohri, F. Pereira, and M. Riley, "The design principles of a weighted finite-state transducer library," *Theoretical Comput. Sci.*, vol. 231, no. 1, pp. 17–32, 2000.

[9] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Comput. Speech Lang.*, vol. 16, no. 1, pp. 69–88, 2002.

[10] T. Hori and A. Nakamura, "Speech recognition algorithms using weighted finite-state transducers," *Synthesis Lectures Speech Audio Process.*, vol. 9, no. 1, pp. 1–162, 2013.

[11] D. Rybach, R. Schlüter, and H. Ney, "A comparative analysis of dynamic network decoding," in *Proc. 2011 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 5184–5187.

[12] D. Nolden, R. Schlüter, and H. Ney, "Search space pruning based on anticipated path recombination in LVCSR," in *Proc. Interspeech*, 2012, pp. 1015–1018.

[13] D. Nolden, R. Schlüter, and H. Ney, "Advanced search space pruning with acoustic look-ahead for WFST based LVCSR," in *Proc. 2013 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 6734–6738.

[14] S. M. Siniscalchi, T. Svendsen, and C.-H. Lee, "A bottom-up modular search approach to large vocabulary continuous speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.,* vol. 21, no. 4, pp. 786–797, Apr. 2013.

[15] G. Zweig and J. Nedel, "Empirical properties of multilingual phone-to-word transduction," in *Proc. 2008 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2008, pp. 4445–4448.

[16] R. G. Wallace, R. J. Vogt, and S. Sridharan,"A phonetic search approach to the 2006 NIST spoken term detection evaluation," in *Proc. Interspeech*, 2007, pp. 2385–2388.

[17] J. Pelemans, K. Demuynck, and P. Wambacq, "A layered approach for dutch large vocabulary continuous speech recognition," in *Proc. 2012 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2012, pp. 4421–4424.

[18] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 369–376.

[19] S. Fernández, A. Graves, and J. Schmidhuber, "Phoneme recognition in TIMIT with BLSTM-CTC," arXiv:0804.3269, 2008.

[20] T. Sainath *et al.*, "Acoustic modelling with CD-CTC-SMBR LSTM RNNS," in *Proc. 2015 IEEE Workshop Autom. Speech Recognit. Understanding*, 2015, pp. 604–609.

[21] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," arXiv:1512.02595, 2015.

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

[23] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," arXiv:1507.08240, 2015.

[24] Y. Miao, M. Gowayyed, X. Na, T. Ko, F. Metze, and A. Waibel, "An empirical exploration of CTC acoustic models," in *Proc. 2016 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 2623–2627.

[25] I. McGraw *et al.*, "Personalized speech recognition on mobile devices," arXiv:1603.03185, 2016.

[26] H. V. Hamme and F. V. Aelten, "An adaptive-beam pruning technique for continuous speech recognition," in *Proc. 4th Int. Conf. Spoken Lang.*, 1996, vol. 4, pp. 2083–2086.

[27] D. Povey *et al.*, "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. Interspeech*, 2016, pp. 2751–2755.

[28] F. Jelinek, L. R. Bahl, and R. L. Mercer, "Design of a linguistic statistical decoder for the recognition of continuous speech," *IEEE Trans. Inf. Theory,* vol. 21, no. 3, pp. 250–256, May 1975.

[29] M. Weintraub, "Keyword-spotting using SRI's DECIPHER large-vocabulary speech-recognition system," in *Proc. 1993 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1993, vol. 2, pp. 463–466.

[30] K. Demuynck, D. Van Compernolle, and H. Van Hamme, "Robust phone lattice decoding," in *Proc. Int. Conf. Spoken Lang. Process.*, 2006, pp. 1622–1625.

[31] D. Moore, J. Dines, M. M. Doss, J. Vepa, O. Cheng, and T. Hain, "Juicer: A weighted finite-state transducer speech decoder," in *Proc. Int. Workshop Mach. Learn. Multimodal Interact.*, 2006, pp. 285–296.

[32] S. J. Young, N. Russell, and J. Thornton, "Token passing: A simple conceptual model for connected speech recognition systems," Cambridge Univ. Eng. Dept., Cambridge, U.K., Tech. Rep. CUED/F-INFENG/TR38, 1989.

[33] P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young, "Large vocabulary continuous speech recognition using HTK," in *Proc. 1994 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1994, vol. 2, pp. 125–128.

[34] A. Stolcke *et al.*, "SRILM-An extensible language modeling toolkit," in *Proc. Interspeech*, 2002, pp. 901–904.

[35] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.,* vol. 15, no. 4, pp. 1352–1365, May 2007.

[36] X. Liu, Y. Wang, X. Chen, M. J. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Proc. 2014 IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 4908–4912.

[37] X. Liu, X. Chen, Y. Wang, M. J. Gales, and P. C. Woodland, "Two efficient lattice rescoring methods using recurrent neural network language models," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 8, pp. 1438–1449, Aug. 2016.

[38] U. V. Chaudhari and M. Picheny, "Improvements in phone based audio search via constrained match with high order confusion estimates," in *Proc. 2007 IEEE Workshop Autom. Speech Recognit. Understanding,* 2007, pp. 665–670.

[39] F. Richardson, M. Ostendorf, and J. R. Rohlicek, "Lattice-based search strategies for large vocabulary speech recognition," in *Proc. 1995 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1995, vol. 1, pp. 576–579.

[40] Y. Zhuang, X. Chang, Y. Qian, and K. Yu, "Unrestricted vocabulary keyword spotting using LSTM-CTC," in *Proc. Interspeech*, 2016, pp. 938–942.

[41] B. Hoffmeister, T. Klein, R. Schlüter, and H. Ney, "Frame based system combination and a comparison with weighted ROVER and CNC," in *Proc. Interspeech*, 2006, pp. 537–540.

[42] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," arXiv:1402.1128, 2014.

[43] Z. Chen, W. Deng, T. Xu, and K. Yu, "Phone synchronous decoding with CTC lattice," in *Proc. Interspeech*, 2016, pp. 1923–1927.

[44] T. Mikolov, S. Kombrink, L. Burget, J. H. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. 2011 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 5528–5531.

[45] P. Le Cerf and D. Van Compernolle, "A new variable frame analysis method for speech recognition," *IEEE Signal Process. Lett.*, vol. 1, no. 12, pp. 185–187, Dec. 1994.

[46] Q. Vu, K. Demuynck, and D. Van Compernolle, "Vietnamese automatic speech recognition: The flavor approach," in *Chinese Spoken Language Processing*. New York, NY, USA: Springer-Verlag, 2006, pp. 464–474.

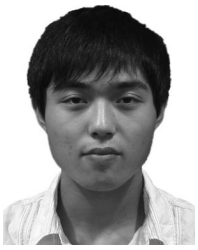
[47] J. Garofalo, D. Graff, D. Paul, and D. Pallett, "Continous Speech Recognition (CSR-I) Wall Street Journal (WSJ0) news, complete," *Linguistic Data Consortium, Philadelphia*, 1993.

[48] J. Nouza and J. Silovsky, "Fast keyword spotting in telephone speech," *Radioengineering*, vol. 18, no. 4, pp. 665–670, 2009.

[49] S. R. Young, "Detecting misrecognitions and out-of-vocabulary words," in *Proc. 1994 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1994, vol. 2, pp. 21–24.

[50] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. 2014 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2014, pp. 4087–4091.

[51] J. Schalkwyk *et al.*, "Your word is my command: Google search by voice: A case study," in *Advances in Speech Recognition*. New York, NY, USA: Springer-Verlag, 2010, pp. 61–90.
[52] J. Droppo and A. Acero, "Context dependent phonetic string edit distance for automatic speech recognition," in *Proc. 2010 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2010, pp. 4358–4361.
[53] C.-Y. Chiang, S. M. Siniscalchi, Y.-R. Wang, S.-H. Chen, and C.-H. Lee, "A study on cross-language knowledge integration in mandarin LVCSR," in *Proc. 2012 8th Int. Symp. Chin. Spoken Lang. Process.*, 2012, pp. 315–319.
[54] M. Lehr and I. Shafran, "Discriminatively estimated joint acoustic, duration, and language model for speech recognition." in *Proc. 2010 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2010, pp. 5542–5545.
[55] N. Shazeer, J. Pelemans, and C. Chelba, "Sparse non-negative matrix language modeling for skip-grams," in *Proc. Interspeech*, 2015, pp. 1428–1432.



Zhehuai Chen (S'14) received the B.S. degree from the Department of Electronic and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2014. He is currently working toward the Ph.D. degree in Shanghai Jiao Tong University, Shanghai, China, working on speech recognition. His current research interests include speech recognition, speech synthesis, and deep learning.

Yimeng Zhuang received the B.S. degree from the Department of Computer Science, Beijing Information Science and Technology University, Beijing, China, in 2014. He is currently working toward the Master degree in Shanghai Jiao Tong University, Shanghai, China, working on keyword spotting. His current research interests include keyword spotting and deep learning.

Yanmin Qian (S'09–M'13) received the B.S. degree from the Department of Electronic and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2007, and the Ph.D. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2012. Since 2013, he has been with the Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU), Shanghai, China, where he is currently an Assistant Professor. From 2015 to 2016, he also worked as an Associate Research in the Speech Group, Cambridge University Engineering Department, Cambridge, U.K. His current research interests include the acoustic and language modeling in speech recognition, speaker and language recognition, key word spotting, and multimedia signal processing.

Kai Yu (SM'10) received the B.Eng. degree in automation, the M.Sc. degree from Tsinghua University, Beijing, China, in 1999 and 2002, respectively, and the Ph.D. degree from Cambridge University, Cambridge, U.K., in 2006 . He then joined the Machine Intelligence Lab. in the Engineering Department, Cambridge University, Cambridge, U.K.. He is currently a Research Professor in Shanghai Jiao Tong University, Shanghai, China. His main research interests include area of speech-based humanCmachine interaction including speech recognition, synthesis, language understanding, and dialogue management. He was selected into the 1000 Overseas Talent Plan (Young Talent) by Chinese government and the Excellent Young Scientists Project of NSFC China. He is a member of the Technical Committee of the Speech, Language, Music, and Auditory Perception Branch of the Acoustic Society of China.