ON LATTICE-FREE BOOSTED MMI TRAINING OF HMM AND CTC-BASED FULL-CONTEXT ASR MODELS

Xiaohui Zhang**, Vimal Manohar***, David Zhang, Frank Zhang, Yangyang Shi,*
Nayan Singhal, Julian Chan, Fuchun Peng, Yatharth Saraf, Mike Seltzer

Facebook AI, USA

ABSTRACT

Hybrid automatic speech recognition (ASR) models are typically sequentially trained with CTC or LF-MMI criteria. However, they have vastly different legacies and are usually implemented in different frameworks. In this paper, by decoupling the concepts of modeling units and label topologies and building proper numerator/denominator graphs accordingly, we establish a generalized framework for hybrid acoustic modeling (AM). In this framework, we show that LF-MMI is a powerful training criterion applicable to both limited-context and full-context models, for wordpiece/mono-char/bi-char/chenone units, with both HMM/CTC topologies. From this framework, we propose three novel training schemes: chenone(ch)/wordpiece(wp)-CTC-bMMI, and wordpiece(wp)-HMM-bMMI with different advantages in training performance, decoding efficiency and decoding time-stamp accuracy. The advantages of different training schemes are evaluated comprehensively on Librispeech, and wp-CTC-bMMI and ch-CTC-bMMI are evaluated on two real world ASR tasks to show their effectiveness. Besides, we also show bi-char(bc) HMM-MMI models can serve as better alignment models than traditional non-neural GMM-HMMs.

Index Terms— LF-MMI, CTC, HMM, modeling units, boost

1. INTRODUCTION

State-of-the-art Automatic Speech Recognition (ASR) systems use Deep Neural Networks (DNN) of various architectures for acoustic modeling (AM). Early success using DNNs for ASR came from hybrid DNN-hidden markov models (DNN-HMM) [1]. These were typically trained using frame-level cross entropy (CE) criterion to predict senones [1] obtained from a previous Gaussian Mixture Model(GMM)-HMM system. Sequence-level training criteria like Maximum Mutual Information (MMI) [2] have been shown to improve the performance of these frame-level trained DNN-HMM-based ASR systems [3, 4]. Since then, various approaches have been shown to be able to train neural network purely through sequence training without initially pre-training using a frame-level criterion – lattice-free MMI (LF-MMI) [5, 6], connectionist temporal classification (CTC) [7], recurrent neural network transducer (RNN-T) [8], attention-based sequence-to-sequence (seq2seq) models [9, 10].

RNN-T and seq2seq models consist of an acoustic encoder that is jointly trained with a neural decoder, which can be considered to be a neural language model (LM). These models can be used to decode audio without using an external LM, and thus can be termed as “end-to-end”. As opposed to this, CTC-based models and hybrid DNN-HMM are “encoder-only” models in the sense that they do not have an explicit jointly trained neural decoder. Having a single “end-to-end” model might be simpler, but in general these models are known to be data-hungry [11, 12] and require thousands of hours of data to achieve competitive performance. RNN-T models are also known to benefit from pre-training encoders or alignments from CTC [13] or hybrid DNN-HMM [14] models for accuracy or efficiency improvements [13, 14, 15, 16]. On the other hand, hybrid models use an external LM for decoding and are often explicitly trained to work with an LM [5, 17, 18]. They are appealing for their modularity which allows to easily replace or extend the lexicon or LM for different applications, while this is still a challenge for end-to-end systems [19]. Hybrid models also explicitly model silence which makes them ideal candidates for pre-processing and segmenting audio as well as for applications that require highly accurate decoding token time-stamps.

While hybrid DNN-HMM and CTC models are very similar, they have vastly different legacies and are usually implemented in very different frameworks. For e.g., though LF-MMI was proposed in a DNN-HMM framework with senone/chenone[20] modeling units, this combination of topology and modeling units is not mandatory. On the other hand, CTC models conventionally refers to a model whose modeling units follow the CTC topology and trained with the Maximum-Likelihood (ML) criteria, which is just the numerator part of the MMI criteria [6]. However, CTC models can also be trained discriminatively with sMBR [21], or MMI [1] criteria. Intrinsically HMM and CTC are just different label topologies (Sec. 3). By decoupling the concepts of modeling units (character/wordpiece/chenone etc.) and label topologies, we introduce a single generalized framework for training hybrid models. This major contribution of our paper allows systematic comparisons (Sec. 6.1) of different modeling units and label topologies to gain deep understandings of their properties, and makes it easier to develop training schemes with novel combinations of them.

From this framework, together with the boost factor [22, 23] for LF-MMI, we propose three new training schemes: 1,2) wp-CTC-bMMI and ch-CTC-bMMI (CTC-bMMI with chenone/wordpiece units), with overall better WERs than HMM-bMMI, whose effectiveness is also confirmed by two real-world server-side/on-device ASR applications. 3) wp-HMM-bMMI, which enables both large-stride (8) inference and accurate token time-stamps, thanks to silence modeling. On the HMM side, we also show HMM-MMI models with bi-character units (bc-HMM-MMI) can serve as a better flat-start trained alignment model than Gaussian Mixture Models (GMM), especially on noisy data.

The authors would like to thank Dan Povey, Yiming Wang, Yiwen Shao, Michael Picheny, and Zhijian Ou for insightful discussions and suggestions.

$\star$ Equal contribution.

[1] The CTC-CRF criterion in [18] is equivalent to LF-MMI as in [6] as both used uniform transition scores constant over the linear chain.

2. LF-BMMI TRAINING

LF-MMI [5] criterion was extended to include boosting [22] in [23, 24]. Here, we present it again in the generalized hybrid model framework for different modeling units and label topologies.

The MMI criterion [2] for training acoustic models can be viewed as maximizing the conditional likelihood of the reference $W^{(r)}$ given the acoustic observation sequence $\mathbf{O}^{(r)}$. This maximizes the joint likelihood of the reference and acoustic observation sequence, i.e. numerator likelihood, and minimizes the marginal likelihood $\mathbf{O}^{(r)}$, i.e. denominator likelihood. As in [5], the denominator is approximated by marginalizing over all state sequences in a denominator graph $\mathcal{G}_{\text{Den}}$ (hence "lattice-free") constructed using an n-gram token LM, which in our case can be phone/character/wordpiece LM. The numerator likelihood is computed by marginalizing over all sequences in a numerator graph $\mathcal{G}_{\text{Num}}(W^{(r)})$ that is similar but constrained to the reference word sequence. In this paper, we assume MMI/bMMI training is always lattice-free, hence omitting "LF" most of the time.

The boosted MMI [22, 3] criterion was introduced to improve training performance by encouraging the criterion to give higher likelihoods to more "accurate" paths. This is achieved by boosting the likelihoods of paths in the denominator graph proportional to the number of errors it contains. The LF-bMMI criterion can be written as:

$$\mathcal{F}_{\text{LF-bMMI}} = \sum_{r} \log \frac{\sum_{\pi \in \mathcal{G}_{\text{Num}}(W^{(r)})} \mathbb{P}\Big(\mathbf{O}^{(r)} \mid \pi\Big)^{\kappa} \mathbb{P}(\pi)}{\sum_{\pi' \in \mathcal{G}_{\text{Den}}} \mathbb{P}(\mathbf{O}^{(r)} \mid \pi')^{\kappa} \mathbb{P}(\pi') e^{-b\mathbb{A}(W^{(r)}, \pi')}}, \tag{1}$$

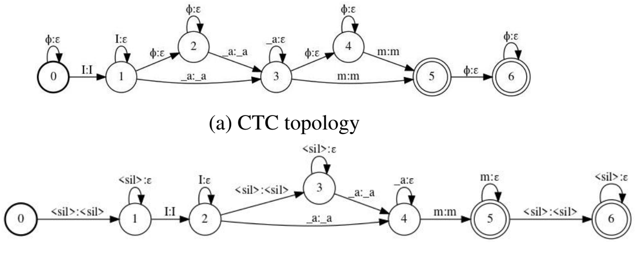
where κ is acoustic weight and $\mathbb{A}(W^{(r)}, \pi')$ is the accuracy function for the path π' measured against the reference $W^{(r)}$. The accuracy function can be defined in several ways such as using phone edit distance to the reference [22]. But implementation-wise, in the lattice-free training framework, it is easiest to define this as a sum of per-frame accuracy values. Therefore, as in [24], we use numerator posterior derived from the numerator graph as a proxy for the per-frame state-level accuracy values. Besides, the intuition of boosted MMI can also be interpreted by Max-Margin learning [25] [26].

2.1. Full-sequence training

The LF-(b)MMI criterion was originally designed at the sequence-level. For efficiency on GPUs, the original Kaldi implementation [5] applies it on equally-sized chunks of around 1.5s each. However, in our application we need to apply LF-bMMI criterion to full-context models like BLSTMs and Transformers, and sequence lengths of up to 2 minutes. We leveraged PyChain's LF-MMI implementation for sequence-training with variable length sequences, and added boosting [22] for training with boosted MMI.

3. LABEL TOPOLOGIES AND MODELING UNITS

In this section, we describe the label topologies and modeling units used in our models. A label topology defines the mapping between a label sequence and neural network output units (i.e. modeling units). For DNN-HMM systems, in this paper, we consider only the 1-state and 2-state-with-skip (which we call as *chain*) HMM topologies [6]. For CTC systems, a CTC topology [7, 18] is used which adds a special blank (ϕ) output unit. The CTC topology defines a mapping $\mathcal{B}^{-1}$ that maps a label sequence $\boldsymbol{l} = l_1, \dots l_L$ to all output unit sequences π such l is obtained by de-duplicating π and removing blank symbols. An intuitive understanding of the difference between CTC and 1-state HMM topology[2] can be obtained by looking at the examples in Fig.1. We see that the CTC topology allows blank (ϕ) units *between any tokens*, e.g. `_a` and `m`. The silence label (`<sil>`) which we see in the 1-state HMM topology is different from blank in that it is a real label similar to any other wordpiece. In our systems, we make the modeling choice to optionally allow it *between words* in order to model the real acoustics of silence [27]. We also point out that explicit silence modeling can help achieve more accurate token time-stamps during decoding, which is an advantage of HMM-based models, especially when time-constraints are used in training targets. Notably, we can use both blank and silence in the same model, which is the case for chenone-CTC models as pointed out in Table 1. On the other hand, we hypothesize that CTC-based models can achieve better training performance as it benefits more from *SpecAugment* due to the blank tokens, verified in our experiments. The blank tokens, which signify "no output" are ideal to represent the perturbations due to feature masking, while HMM-based models are forced to model masked features using non-silence units (except between words where silence can be predicted). However the cost is less accurate decoding time-stamps due to the peaky behavior [28] caused by dominance of blank tokens at output during decoding.



(a) CTC topology

(b) 1-state HMM topology

Fig. 1: Numerator FSTs (mapping output units to modeling units) of 'I', '_a', 'm' in CTC and 1-state HMM topology; ϕ means blank.

We consider the following 4 types of labels:

- Mono-character (mono-char): This is the simplest case where the labels are characters which are context-independent.
- Bi-character (bi-char): In this case, the characters are modeled separately for each left-context. We do a basic text-based clustering based on the raw counts of the character n-grams seen in the training transcripts, to let infrequent bi-characters share a modeling unit within each cluster.
- Tri-character (tri-char): In this case, the characters are modeled separately for each left and right context. We use standard decision-tree based clustering of states [29] and share modeling units across states within each cluster. We refer to tri-char based modeling units as chenones [20].
- Wordpiece (wp): In this case, wordpieces are constructed using Sentencepiece [30] modeling from training transcripts.

4. NUMERATOR AND DENOMINATOR PREPARATION

4.1. HMM topology

Chenone units: We use the approach for denominator graph preparation from [5], except for replacing phoneme with character i.e. by composing an $\{3,4\}$-gram character LM with tri-character context-dependency transducer and HMM transducer. The n-gram LM was

[2] The *chain* topology can be obtained from Fig.1b by replacing input tokens on all self-loops by a '2nd version' of each token (e.g. 'I'→'I_2').

estimated using the alignments from a previous flatstart trained hybrid LFMMI bi-char system [6]. Numerator graph preparation also follows the same approach from [5] and we apply time-constraints using alignments from the same flatstart-trained hybrid system.
Bi/mono-char units: The denominator graph preparation follows the similar approach as described for chenone units in the previous section but using a bi-character context-dependency for the bi-char systems and no context-dependency for the mono-char systems. Also the character LM has to be estimated from transcripts rather than alignments, with randomly inserted silence phones [6]. Numerator graph in this case is a full HMM with self-loops following [6].
Wordpiece units: The denominator and numerator graph preparation mimic the approach for mono-char units as described in the previous section. The word sequences are converted to wordpieces using a "wordpiece lexicon" constructed using mappings from a Sentencepiece model [31] trained on the text. Since the number of wordpieces is usually much larger than the number of characters, to decrease the denominator graph size, we use a $\{2,3\}$-gram LM on wordpieces for the denominator. An example for numerator FST for wordpiece units with HMM topology is shown in Figure 1b.

4.2. CTC topology

Chenone units: For chenone units with CTC topology, we first obtain the chenone sequence from a previous flatstart trained hybrid LFMMI bi-char system as in the case of HMM topology described in the chenone-HMM case, and remove repetitions to obtain a label sequence with chenones as the labels. We treat chenones similar to regular characters and compose the sequence with the CTC topology transducer [18]. Note that unlike the chenone HMM case, there's no time constraints on numerator FSTs here. For the denominator graph, we first obtain the denominator graph for a 1-state HMM topology as in HMM case, and then convert it to a CTC compatible topology by splitting each state into two states and adding two arcs for consuming blank tokens, in the same way as done in [32] for constructing decoding graph for chenone-based CTC models.
Wordpiece units: For wordpiece units with CTC topology, the numerator graph (e.g. 1b) is created in the same way as the chenone case. The wordpiece sequence is generated on-the-fly by tokenizing the reference sequence into wordpieces using a SentencePiece model [31]. The denominator graph is created by composing a n-gram wordpiece LM with the CTC topology transducer. The n-gram wordpiece LM is estimated from training transcripts tokenized into wordpieces using a Sentencepiece model.

We summarized the main properties of the combinations of HMM/CTC topology with different modeling units which we'll study in Table 1. Among them, wp-HMM, wp-CTC and ch-CTC are novel schemes in terms of MMI training.

5. PRE-TRAINING WITH CE/ML MODELS

To improve LF-bMMI training performance, we can pre-train the model with either frame-level CE criterion or sequence level ML criterion [6][3]. ch-HMM models (i.e. HMM topology with chenone units) are the only one for which we use frame-level alignments. For these, we use CE pre-training with the labels obtained from frame-level alignments. For other models, we use sequence-level pre-training with ML criterion. Note that in the case of CTC topology, this is equivalent to the CTC training criterion. In all these cases, the neural network outputs are locally normalized by a softmax layer.

When fine-tuning a neural network pre-trained with CE or ML criterion, we empirically found removing softmax and using the logits directly helped performance. However, we subtract the log of the model priors from the logits just as we would when using the model for decoding [1]. We estimate the model priors [33] on a small subset of training data as opposed to the conventional approach of obtaining it from frame-level alignments [1]. This approach is more general as it allows to estimate model priors even for CTC-based systems with blank tokens and for wordpiece-based systems. We additionally apply an acoustic scale κ on the neural network outputs before it is combined with the graph scores from the numerator or denominator graphs. In theory, the LF-bMMI objective is normalized at the sequence-level and hence it is capable of learning the linear offset corresponding to the log-priors as well as the acoustic scale. We indeed find that when the model is trained from scratch, we do not need to explicitly supply the log priors or an acoustic scale of 1.0 suffices. But when fine-tuning a pre-trained network, we found that we need to match the priors and acoustic scale to the optimal values during decoding. Using a mis-matched prior or acoustic scale leads to slower convergence.

6. EXPERIMENTS

6.1. Comprehensive Analysis on Librispeech

Here we perform a series of analysis of LF-bMMI training with different modeling units, label topologies and various configurations on Librispeech [34]. We use the standard (960h) training and (*dev-clean*, *dev-other* sets for training and evaluation respectively. We use the official 4-gram LM pruned to 3-gram with a threshold of $1e^{-9}$) built into HLG/HCLG graphs for decoding. For the AM, we use a 25M-parameters TDNN-BLSTM network with 2 BLSTM [35] layers (640 hidden units) in each recurrence direction and 3 TDNN layers [36, 37] (640 hidden units) interleaved between input and first BLSTM layer, and between the 2 BLSTM layers. Unless specified, we use stride (i.e. input frame rate / output frame rate) 8 for wp-CTC/HMM and stride 4 for ch-CTC/HMM models, since previous studies [32] have shown wordpieces units can work reasonably well with stride 8, while chenone units cannot because of their short duration. Regarding modeling units, for mc-HMM, we use 29 characters. For bc-HMM, we use 870 bi-char units from text-based clustering. For ch-HMM/ch-CTC systems, we use a set of 1632 chenones corresponding to a tree built from alignments from a bc-HMM model. For wp-HMM/wp-CTC systems, we use a set of 511 wordpieces built from a Sentencepiece model, balancing performance between strides 4 and 8. Unless specified, we always conduct MMI training without pre-training, with 0 as the boost factor, `LD` as the *SpecAugment* policy, and 1-state topology for HMM-based systems.

6.1.1. Basic results and the effect of ML/CE pre-training

We first do comparison of the WERs of LF-MMI training for wp-HMM/CTC and ch-HMM/CTC with their corresponding non-discriminatively trained ML/CE baselines, and then investigate the effect of pre-training with ML/CE for LF-MMI training. Regarding the choice between ML/CE training, since ch-HMM is the only one with frame-level targets, it's natural to go with CE for ch-HMM, and ML for others. From results in Table 2, comparing with ML baselines, we can see that wp/ch-CTC-MMI both have around $8-15\%$ relative improvements on *dev-other* and $4-7\%$ relative improvements on *dev-clean*, and pre-training MMI with ML helps provide a better initialization resulting in both faster convergence and better final WER. For wp-HMM, the ML WER is significantly worse and doesn't help for pre-training MMI, which is similar to the finding on

[3] Strictly speaking CE is frame-level ML. We make CE comparable to ML since we always refer ML to "sequence-level ML" in our paper for simplicity.

Table 1: Properties of combinations of different modeling units and label topologies ('mc' = 'mono-char', 'bc' = 'bi-char', 'ch'='chenone')

Model	wp-HMM	mc/bc-HMM	ch-HMM	ch-CTC	wp-CTC
Label topology	HMM			CTC	
Acoustic-based clustering	N		Y		N
Time-constrained Num. FST	N		Y	N	
Explicit silence modeling	Y				N
Training criterion	ML / MMI		CE / MMI	ML / MMI	

Table 2: *dev-clean/other* ML/CE vs. MMI WER and the effect of ML/CE pre-training for MMI (#ep means # epochs to reach the best WER).

Loss	wp-HMM		wp-CTC		ch-CTC	
	WER	#ep	WER	#ep	WER	#ep
ML	7.2 / 17.3	69	4.6 / 11.5	58	4.1 / 10.7	55
MMI	4.3 / 11.0	60	4.4 / 10.6	121	3.8 / 9.1	153
ML → MMI	4.4 / 11.0	66	4.1 / 10.2	89	3.7 / 9.0	143

Loss	ch-HMM	
	WER	#ep
CE	4.2 / 10.6	60
MMI	4.0 / 9.5	54
CE → MMI	3.8 / 9.1	48

mc-HMM in [6]. For ch-HMM, MMI achieves $5 - 10\%$ improvement comparing with CE, and pre-training with CE further brings 4% improvements.

6.1.2. The effect of boost

Here we study the contribution of the boost factor for bMMI training. From Table 3 we can see that the boost improves WERs for all four systems. For wp-HMM, wp-CTC, ch-CTC, the relative WER gain is around $2 - 7\%$ on *dev-clean* and $2 - 4\%$ on *dev-other*. For ch-HMM, the gain is large: 10% on *dev-clean* and 6% on *dev-other*. We suspect the reason is that ch-HMM is the only system with time-constraints on the numerator FSTs, and thereby the frame posteriors are more accurate, which the boosting mechanism relies on.

Table 3: *dev-clean/other* bMMI WER with different boost values

boost	wp-HMM	wp-CTC	ch-HMM	ch-CTC
0	4.3 / 11.0	4.4 / 10.6	4.0 / 9.5	3.8 / 9.1
0.3	4.2 / 11.0	**4.2 / 10.4**	3.7 / 9.2	3.7 / 9.1
0.5	**4.2 / 10.7**	4.3 / 10.3	**3.6 / 8.9**	**3.6 / 8.7**
1.0	4.2 / 10.9	4.4 / 10.9	3.6 / 9.3	3.6 / 8.9

6.1.3. The effect of SpecAugment

Here we study the effect of *SpecAugment* for different systems. We study two *SpecAugment* policies – LD, Large. LD is same as in [38] but with maximum time mask width of $p = 0.2$. Large ($T = 30, mT = 10$) is a more aggressive policy which was shown in [32] to help performance on Librispeech. From Table 4, we see that without *SpecAugment*, for both wordpiece and chenone units, HMM and CTC models have similar WERs. However, we see that CTC models benefit more from *SpecAugment* compared to the corresponding HMM models, verifying our hypothesis on the advantage of CTC which better models feature masking with blank tokens.

Table 4: *dev-clean/other* MMI WERs with different *SpecAugment* policies

Policy	wp-HMM	wp-CTC	ch-HMM	ch-CTC
None	4.8 / 13.0	4.8 / 13.1	4.5 / 11.6	4.5 / 11.7
LD	4.3 / 11.0	4.4 / 10.6	4.0 / 9.5	3.8 / 9.1
Large	4.4 / 10.8	4.3 / 10.3	3.9 / 9.2	3.8 / 8.9

6.1.4. Comparing different modeling units

Here we fix the label topology to be 1-state HMM, and compare the WER and RTF[4] performance of different modeling units, both wordpiece and character-based units. For wordpiece, we train models with strides 8 and 4. For character-based units we couldn't get reasonable convergence performance with stride 8 and hence stick to stride 4. From Table 5, we can see that the WER of bi-char is better than mono-char by a large gap ($13 - 15\%$ relative), while the relative improvement of tri-char on top of bi-char is smaller ($2 - 7\%$). This implies that even text-based simple clustering can provide quite useful context dependency information. Looking at wordpiece units, we can see that with stride 4, its performance is better than bi-char and close to tri-char, showing wordpieces can also be powerful modeling units without relying on decision tree building. Furthermore, at stride 8, we can see its performance is still $6 - 10\%$ better than mono-char at stride 4. Unfortunately, the RTFs we report here for wordpiece-based models are much worse than the mono-char case. This is due to increased number of modeling units (29 chars → 511 wordpieces), and hence more confusable paths during graph search. However, in real applications where we use much larger AMs so that AM inference dominates the computation, the RTF advantage of stride 8 wordpiece systems would amplified as verified in a previous study [12], where a stride 8 wp-CTC model had better RTF than a stride 3 ch-HMM model using the same encoder. We also measure decoding time-stamp accuracy of different models. The metric is the mean absolute error (MAE) between the start/end time-stamps of decoded hypothesized words and reference words, with incorrect words ignored. The reference time-stamps were obtained by aligning the audio with the reference using a bc-HMM system. In table 5, we report this metric as time-stamp-error (TSE, in ms) on *dev-other*.

Table 5: *dev-clean/other* MMI WER, RTF and TSE of different units with the same (1-state) HMM topology

Unit	wordpiece		mono-char	bi-char	chenone
Stride	8	4			
WER	4.4 / 11.1	3.9 / 10.1	4.9 / 11.8	4.2 / 10.3	4.1 / 9.6
RTF	0.020	0.046	0.006	0.005	0.011
TSE	86	66	74	47	28

[4] When we measure RTF, we optimize the decoding beam so that the WER is 1% worse than the optimal WER. Otherwise we always use a beam of 30.

We see that the ch-HMM model has the smallest TSE, confirming time-constraints in training targets helps the model to learn more accurate alignments.

6.1.5. The effect of HMM topology

Here we compare the impact of 1-state vs *chain* HMM topology for wp-HMM and ch-HMM models. For wp-HMM, in the *chain* case, the set of modeling units gets doubled from the 511 wordpieces as in the 1-state case. For ch-HMM, we choose a 3008-sized tree for the *chain* case, which is around two times of the 1632-leaves tree for the 1-state case. From Table 6, we can see the impact on ch-HMM models is minor. However the impact on wp-HMM is obvious on *dev-other*, where *chain* topology brings 5% WER gain, which agrees with the finding in [6]. We believe the reason behind the observation is that: The richer representation provided by *chain* topology, better modeling intra-class variations, contributes more to wordpiece units which are longer than chenones.

Table 6: *dev-clean/other* MMI WER of wp-HMM and ch-HMM with 1-state and *chain* HMM topology

	wp-HMM		ch-HMM	
Topo.	1-state	*chain*	1-state	*chain*
WER	4.3 / 11.0	4.3 / 10.5	4.0 / 9.5	4.0 / 9.4

6.1.6. The effect of denominator LM order

Here we investigate the impact of denominator LM order on denominator FST size and training speed for wordpiece/chenone systems (wp-CTC/ch-HMM). From Table 7 we can see that due to a large set of units which the den. LM is built upon, and the large CTC topology transducer (For a reference, den. FST w/ a 3-gram den. LM for wp-HMM is 5.2M), den. FST size in the wordpiece case is much larger than the chenone case, so that when increasing the order from 2 to 3, per-epoch training time increased by 110%, while it only increases by 12% when changing order from 3 to 4 for ch-HMM. In terms of total training time, when increasing den. LM orders, wp-CTC training becomes much more expensive, while ch-HMM training even becomes cheaper. Considering the WER improvement for wp-CTC still looks worthwhile, we decide to stick with order 3 for wordpiece systems and 4 for chenone systems in other experiments.

6.1.7. Benchmarking the 4 main systems with their optimal setup

Here we conduct a comprehensive WER/RTF/TSE benchmark of the 4 main systems we have studied: wp-HMM, wp-CTC, ch-HMM, ch-CTC with their optimal training setup: optimal boost value for each, *SpecAugment* `Large` policy for all, pre-training for all except wp-HMM, *chain* topology for wp/ch-HMM. From Table 8, we can see as expected, ch-HMM achieves the best TSE performance thanks to silence modeling and time-constraints used in training, ch-CTC achieves the best WER (thanks to blank+*SpecAugment*), and also RTF. For wp-HMM and wp-CTC, they perform similarly well on RTF/WER (with wp-CTC's WER at stride 4 being a bit better), while wp-HMM's TSE is much better again thanks to silence modeling. This shows that wp-HMM, which doesn't rely on alignments, is an appealing choice when we need a large-stride & flat-start trained model providing accurate timestamps. Besides, though ch-CTC has worse TSE than ch-HMM (due to lack of time-constraints in training and CTC's peaky behavior), the gap is much smaller than that of wp-CTC/HMM, showing that silence modeling (which ch-CTC has but wp-CTC doesn't) can effectively improve time-stamp accuracy, even for CTC-based models.

Table 7: *dev-clean/other* MMI WER, denominator LM order/FST size, and training speed for wp-CTC and ch-HMM

	wp-CTC		ch-HMM	
den. LM order	2	3	3	4
den. FST size	4.2MB	10.2MB	3.8MB	4.6MB
WER	4.8 / 11.4	4.4 / 10.6	4.3 / 9.9	4.0 / 9.5
# epochs	112	121	84	54
per-epoch hrs	0.38	0.8	1	1.12

Table 8: *dev-clean/other* bMMI WER/RTF/TSE of optimal systems

	wp-HMM		wp-CTC		ch-HMM	ch-CTC
Stride	8	4	8	4		
WER	4.0/10.1	3.9/9.7	4.0/10.1	3.7/9.4	3.5/8.5	3.3/8.3
RTF	0.023	0.053	0.027	0.052	0.015	0.011
TSE	59	45	162	112	25	51

6.2. CTC-bMMI training for real-world large-scale ASR tasks

Here we apply the proposed CTC-bMMI training scheme with wordpiece/chenone units (i.e. wp-CTC-bMMI and ch-CTC-bMMI) in two real world large scale ASR tasks and compare with the corresponding ML baselines to confirm its effectiveness. In the first application, we adopt wp-CTC-bMMI for training a large full-context Transformer model, for server-side ASR. In the second application, we adopt ch-CTC-bMMI for training a small limited-context streamable[5] Emformer [39] using convolution operations similar to Conformer [40], for on-device ASR. We focus on CTC-bMMI rather than HMM-bMMI because the emphasis in the applications here is on WER rather than the token time-stamp accuracy.

6.2.1. wp-CTC-bMMI for training large Transformer models

Here, we compare CTC-bMMI with the standard CTC (i.e. CTC-ML) and RNN-T criteria on a real-world large scale English video ASR task. The training data consist of de-identified public videos with no personal identifiable information (PII), where only the audio part is used. Besides a development set, there are 3 test sets under different audio conditions: *clean*, *noisy* and *extreme*. These test sets are further segmented by into audio chunks that are no longer than 45 seconds. Decoding is performed on these chunks unless otherwise specified. Training data are segmented into chunks with a maximum duration of 10s. Besides 39.4K hours of supervised training data (including two speed perturbed copies), we prepared 2.2M hours of unsupervised training data, with transcriptions obtained by decoding de-identified public videos by our internal ASR models. Several data filters are applied to keep the most useful data, e.g. confidence filter, word-per-second filter and country filter, etc. No human effort is involved in transcribing these unsupervised data. In total, we have 1.5M hours of semi-supervised training data.

We use the same Transformer encoder architecture for each model, consisting of 24 layers, each with 12 attention heads, 768 embedding dimensions, and 3072 feed-forward dimensions. The encoder part has roughly 170M parameters. The input is the same as all other experiments: 80-dimensional log-Mel filter bank features at a 10ms frame rate. A stride of 8 is applied at the input layer by

[5] Though the emphasis of our paper is bMMI for full-context ASR model training, we intentionally choose a limited-context scenario to show our method can work for streamable models as well.

concatenating every 8 feature frames and then project to a dimension of 768, the same as the Transformer embedding dimension. For the RNN-T model, a predictor network consists of 512-dimensional embeddings for each token followed by two LSTM layers with 512 hidden nodes, then a linear projection to 1024-dimensional features before the joiner. For the joiner, the combined embeddings from the encoder and the predictor first go through a $tanh$ activation and then another linear projection to the target number of wordpieces. We use the same set of 511 wordpieces as modeling units for all models, and use the same 4-gram LM for decoding CTC and CTC-bMMI models. In improve help convergence, for CTC(-ML) training we used CTC loss at intermediate layers. For RNN-T training we used CE loss at intermediate layers. The CTC-bMMI model is pre-trained by the CTC model. The boost value used is 2. Experiment results could be found in Table 9. We see that the CTC-bMMI model has large ($4-7\%$) WER improvements over CTC especially on the *noisy* and *extreme* sets and is almost on-par with the RNN-T model even without neural LM rescoring. RTF-wise, all models are similar.

Table 9: Comparing training criteria for Transformer-based ASR

Loss	*clean*	*noisy*	*extreme*	RTF
CTC	8.53	12.10	18.46	0.089
CTC-bMMI	8.24	11.61	17.19	0.090
RNN-T	8.01	11.49	17.04	0.094

6.2.2. *ch-CTC-bMMI for training small Emformer models*

Here we study the effectiveness of CTC-bMMI using chenone modeling units in an on-device English ASR scenario, with CTC(-ML) as baselines. Training data are two subsets of the data used in Sec. 6.2.1, containing 7000 and 1000 hours of videos correspondingly. We use the same test data as in Sec. 6.2.1. The model is an Emformer [39] model supporting streaming speech recognition using block processing. In training, attention mask and "right context hard copy" are used to constrain the look ahead context for self-attention. In this experiment, each block consists of 1.4 seconds left context, 600 ms center chunk size, and 40 ms look-ahead context size. The algorithmic latency [39] of the acoustic model is 340 ms. A stride of 4 is applied at the input layer by concatenating every 8 feature frames and then project to a dimension of 256, used as input to the stack of 12 Emformer layers. Each Emformer layer has a multi-head self-attention layer with four heads, input size 256, a feed-forward layer with hidden dim 1024, and a depth separable convolution layer with kernel size 15. The model has roughly 18M parameters. From the results in Table 10, we can see that in the 1000h condition, CTC-bMMI has $20-30\%$ relative WER improvement over CTC, which is much larger than the gain ($11-16\%$) in the 7000h condition, showing discriminative training helps more when we have less data, and when the models are smaller (comparing with Table 9).

Table 10: Comparing training criteria for Emformer-based ASR

Criterion	*clean*	*noisy*	*extreme*	training hours
CTC	25.38	32.18	39.65	1000h
CTC-bMMI	17.63	23.36	31.02	
CTC	18.44	23.97	31.38	7000h
CTC-bMMI	15.45	20.59	27.71	

6.3. bc-HMM-MMI for alignment model training

Here, we study an important application of HMM-MMI models with bi-char units (bc-HMM-MMI): alignment generation. Accurate alignments are important for ASR, in terms of both audio segmentation and providing training targets for main/auxiliary ASR training tasks even for RNN-T [14, 15]. In order to train an alignment model from scratch, people have been mainly relying on GMM-HMMs, e.g. from Kaldi [41]. However, single-stage trained, HMM-based neural models, e.g. bc-HMM-MMI models, can be more appealing candidates (used in Kaldi OCR recipes[42] already), which may provide more accurate alignments especially on noisy data, and moreover, enable an all-neural acoustic modeling pipeline. To the best of our knowledge, there's no prior literature confirming this by benchmarking bc-HMM-MMI models with GMM-HMMs. Here we conduct this benchmark by training a bc-HMM-MMI neural model and a GMM model (following the Kaldi recipe) with the same data and graphemic lexicon, evaluate their WERs, and then generate alignments on the same training data, on top which we then train two CE neural models and evaluate their WERs for measuring the alignment quality. The two CE models and the bc-HMM-MMI alignment model all have the same architecture as the one used in 6.1, except the stride is 3 here. Using the same architecture enables us to show another advantage of bc-HMM-MMI alignment models: Besides generating alignments, it can also serve as a pre-trained seed model for the following modeling stage to improve training performance, which can't be done with GMMs. We conduct the experiments on Librispeech where we train models on the full 960h data and evaluate WERs on *dev-other*, and a Tagalog Video ASR task (whose description is the same as 6.2.1) where we train models on 1000h Tagalog videos and evaluate WERs on the *noisy* test set. From the results shown in Table 11, we can see that the bc-HMM-MMI neural alignment model achieves on-par alignment quality as GMM evaluated by CE WER. On Tagalog Video ASR, which is much noiser than Librispeech, the bc-HMM-MMI model is capable of generating much better alignmetns, reducing CE WER by 14% relatively. Besides, pre-traing the CE models with bc-HMM-MMI seed models indeed bring down CE WERs further, by 2% (Tagalog) or 5% (Librispeech) relatively. This shows besides serving as a strong alignment model, a bc-HMM-MMI model can also serve as as a seed model for downstream modeling tasks.

Table 11: Alignment model and CE model WERs, on Tagalog video (*noisy*) and Librispeech (*dev-other*)

	Alignment Model		CE	CE w/ seed
Tagalog Video	GMM	61.7	38.0	-
	bc-HMM-MMI	27.5	32.6	31.9
Librispeech	GMM	30.1	11.3	-
	bc-HMM-MMI	10.0	11.2	10.6

7. CONCLUSION

In this paper, we generalized the original chunk-wise HMM-based LF-bMMI training framework to a new framework, where full-context neural network training is enabled by full-sequence LF-bMMI training, supporting both HMM and CTC as the label topology, and mono-char/bi-char/chenone/wordpieces as modeling units. Comprehensive studies were conducted on Librispeech to understand the impact of boost factor, CE/ML pre-training, *SpecAugment* and denominator LM order to different training schemes. From this framework, we proposed wp-CTC-bMMI and ch-CTC-bMMI training schemes with WER advantages, studied also in two large scale real-world ASR tasks, and wp-HMM-bMMI training scheme with advantages in large-stride inference, time-stamps accuracy, and alignment-free training. In the future we would like to further generalize LF-bMMI training to RNN-T-type of topologies.

8. REFERENCES

[1] George E. Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[2] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum mutual information estimation of hidden markov model parameters for speech recognition," in *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1986, vol. 11, pp. 49–52.

[3] K. Vesely, M. Hannemann, and L. Burget, "Semi-supervised training of deep neural networks," in *ASRU 2013*.

[4] George Saon and Brian Kingsbury, "Discriminative feature-space transforms using deep neural networks," in *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[5] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free mmi," in *Interspeech*, 2016.

[6] Hossein Hadian, Hossein Sameti, Daniel Povey, and Sanjeev Khudanpur, "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.

[7] Alex Graves, Santiago Fernández, and Faustino Gomez, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ICML 2006*.

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

[9] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philémon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4945–4949.

[10] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[11] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4774–4778.

[12] Xiaohui Zhang, Frank Zhang, Chunxi Liu, Kjell Schubert, Julian Chan, Pradyot Prakash, Jun Liu, Ching-Feng Yeh, Fuchun Peng, Yatharth Saraf, and Geoffrey Zweig, "Benchmarking lf-mmi, ctc and rnn-t criteria for streaming asr," in *2021 IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 46–51.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[14] Chunxi Liu, Frank Zhang, Duc Le, Suyoun Kim, Yatharth Saraf, and Geoffrey Zweig, "Improving rnn transducer based asr with auxiliary tasks," in *2021 IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 172–179.

[15] Jay Mahadeokar, Yuan Shangguan, Duc Le, Gil Keren, Hang Su, Thong Le, Ching-Feng Yeh, Christian Fuegen, and Michael L Seltzer, "Alignment restricted streaming recurrent neural network transducer," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 52–59.

[16] Albert Zeyer, André Merboldt, Ralf Schlüter, and Hermann Ney, "A New Training Pipeline for an Improved Neural Transducer," in *Proc. Interspeech 2020*, 2020, pp. 2812–2816.

[17] V. Manohar, H. Hadian, D. Povey, and S. Khudanpur, "Semi-supervised training of acoustic models using Lattice-Free MMI," in *ICASSP*, 2018.

[18] Hongyu Xiang and Zhijian Ou, "Crf-based single-stage acoustic modeling with ctc topology," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5676–5680.

[19] Duc Le, Mahaveer Jain, Gil Keren, Suyoun Kim, Yangyang Shi, Jay Mahadeokar, Julian Chan, Yuan Shangguan, Christian Fuegen, Ozlem Kalinli, Yatharth Saraf, and Michael L. Seltzer, "Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion," *CoRR*, vol. abs/2104.02194, 2021.

[20] Duc Le, Xiaohui Zhang, Weiyi Zheng, Christian Fügen, Geoffrey Zweig, and Michael L Seltzer, "From senones to chenones: Tied context-dependent graphemes for hybrid speech recognition," *ASRU*, 2019.

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

[22] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah, "Boosted mmi for model and feature-space discriminative training," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 4057–4060.

[23] Zhehuai Chen, Yanmin Qian, and Kai Yu, "Sequence discriminative training for deep learning based acoustic keyword spotting," *Speech Communication*, vol. 102, pp. 100–111, 2018.

[24] Chao Weng and Dong Yu, "A comparison of lattice-free discriminative training criteria for purely sequence-trained neural network acoustic models," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6430–6434.

[25] Murali Karthick Baskar, Lukáš Burget, Shinji Watanabe, Martin Karafiát, Takaaki Hori, and Jan Honza Černockỳ, "Promising accurate prefix boosting for sequence-to-sequence asr," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5646–5650.

[26] Kevin Gimpel Noah A Smith, "Softmax-margin training for structured log-linear models," .

[27] Guoguo Chen, Hainan Xu, Minhua Wu, Daniel Povey, and Sanjeev Khudanpur, “Pronunciation and silence probability modeling for asr,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[28] Albert Zeyer, Ralf Schlüter, and Hermann Ney, “Why does ctc result in peaky behavior?,” *arXiv preprint arXiv:2105.14849*, 2021.

[29] S. J. Young, J. J. Odell, and P. C. Woodland, “Tree-based state tying for high accuracy acoustic modelling,” in *Proceedings of the Workshop on Human Language Technology*, USA, 1994, HLT ’94, p. 307–312, Association for Computational Linguistics.

[30] Taku Kudo and John Richardson, “SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Brussels, Belgium, Nov. 2018, pp. 66–71, Association for Computational Linguistics.

[31] Taku Kudo and John Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” *arXiv preprint arXiv:1808.06226*, 2018.

[32] Frank Zhang, Yongqiang Wang, Xiaohui Zhang, Chunxi Liu, Yatharth Saraf, and Geoffrey Zweig, “Faster, Simpler and More Accurate Hybrid ASR Systems Using Wordpieces,” in *Proc. Interspeech 2020*, 2020, pp. 976–980.

[33] Vimal Manohar, Daniel Povey, and Sanjeev Khudanpur, “Semi-supervised maximum mutual information training of deep neural network acoustic models,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[34] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An asr corpus based on public domain audio books,” in *ICASSP*, 2015.

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

[36] Kevin J Lang, Alex H Waibel, and Geoffrey E Hinton, “A time-delay neural network architecture for isolated word recognition,” *Neural networks*, vol. 3, no. 1, pp. 23–43, 1990.

[37] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[38] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, “Specaugment: A simple data augmentation method for automatic speech recognition,” *arXiv preprint arXiv:1904.08779*, 2019.

[39] Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, and Others, “Emformer: Efficient Memory Transformer Based Acoustic Model For Low Latency Streaming Speech Recognition,” in *Proc. ICASSP*, 2021.

[40] Anmol Gulati, James Qin, Chung Cheng Chiu, and Others, “Conformer: Convolution-augmented transformer for speech recognition,” in *Proc. INTERSPEECH*, 2020.

[41] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, “The kaldi speech recognition toolkit,” in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. Dec. 2011, IEEE Signal Processing Society, IEEE Catalog No.: CFP11SRW-USB.

[42] Ashish Arora, Chun Chieh Chang, Babak Rekabdar, Bagher BabaAli, Daniel Povey, David Etter, Desh Raj, Hossein Hadian, Jan Trmal, Paola Garcia, et al., “Using asr methods for ocr,” in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 663–668.