# Less Peaky And More Accurate CTC Forced Alignment by Label Priors

Ruizhe Huang[1†], Xiaohui Zhang[2], Zhaoheng Ni[2], Li Sun[5†], Moto Hira[2], Jeff Hwang[2], Vimal Manohar[2], Vineel Pratap[2], Matthew Wiesner[1], Shinji Watanabe[3], Daniel Povey[4], Sanjeev Khudanpur[1]

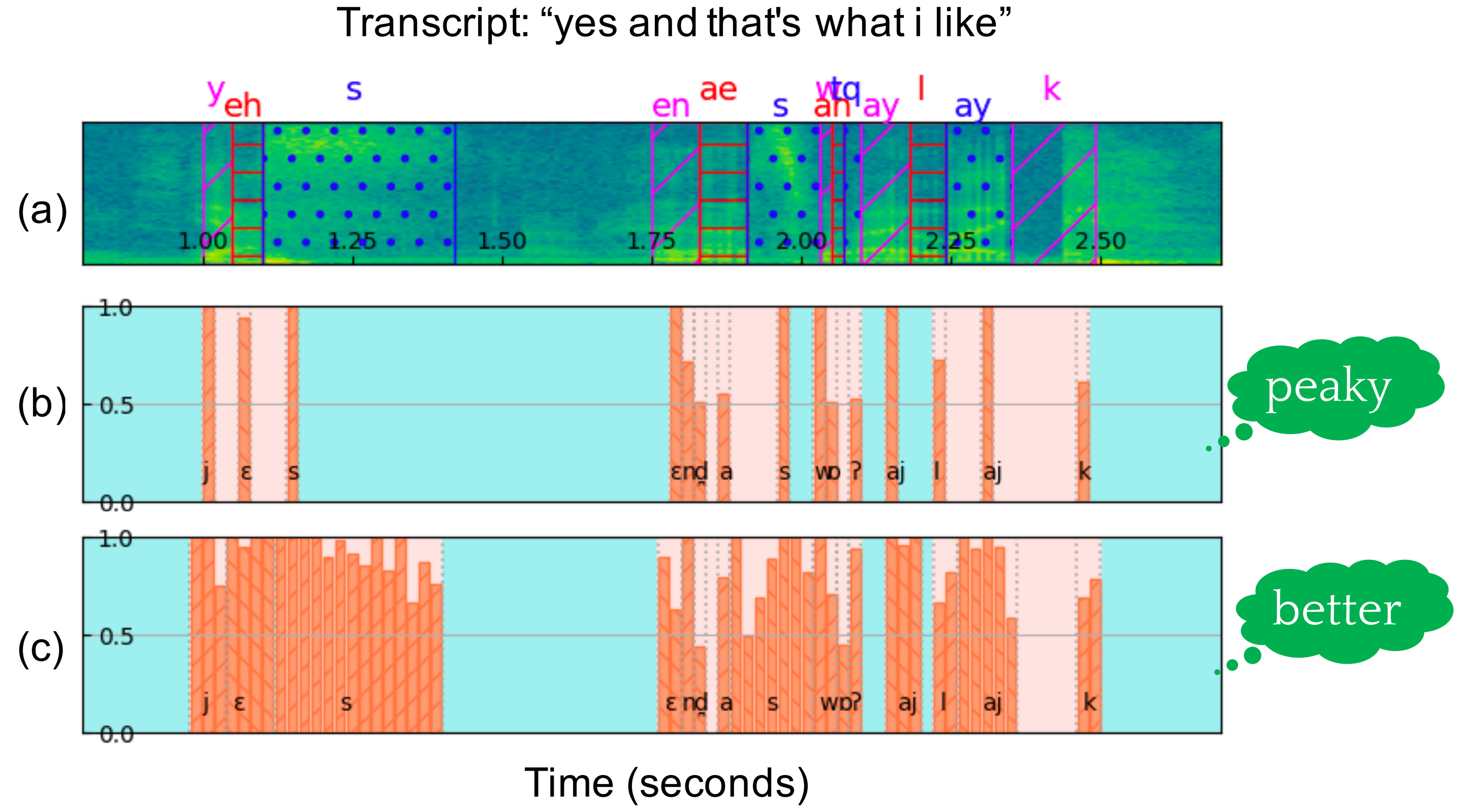
[1]Johns Hopkins University, USA [2]Meta AI, USA [3]Carnegie Melon University, USA [4]Xiaomi Corp., Beijing, China [5]Boston University, USA

[†]Work done during internship at Meta

Contact information: ruizhe@jhu.edu zni@meta.com

## TL;DR

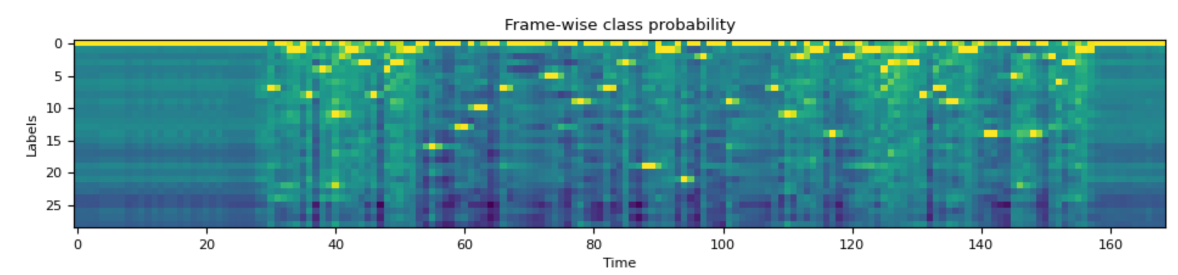
In this work, we found that the **peaky behavior of CTC models** can be alleviated by applying **label priors** during training. This can generate **more accurate forced alignment timestamps** than standard CTC.




## Methodology

This is done simply by dividing the CTC posterior probabilities with the label priors during training.

This is a posteriorogram for an utterance from a CTC model




**Accumulate** label priors for each label by summing over the (unscaled) rows

**Divide** each row of the posteriors by its label priors from last epoch. Then compute CTC loss

Note: we found the CTC loss implemented in PyTorch doesn't support this. Please check out the issue #122243 in PyTorch repo. We used k2 library to compute CTC loss instead.

$$P_{\text{with\_priors}}(\pi|\mathbf{X}) = \prod_{t=1}^{T} y^t_{\pi_t} / P(\pi_t)^\alpha$$

- $P(\pi|X)$: the CTC posterior of a path
- $\pi$: an alignment path
- $\pi_t$: the label at time t in the path
- $y^t_{\pi_t}$ : the CTC posterior for label $\pi_t$ at time t
- $P(\pi_t)$ : the label prior for label $\pi_t$
- $\alpha$: a scaling factor

$$P_{\text{ctc\_with\_priors}}(\mathbf{W}|\mathbf{X}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{W})} P_{\text{with\_priors}}(\pi|\mathbf{X})$$

Sum over all paths with dynamic programming
$\mathbf{W}$: a word sequence, which can have many alignment paths
$\bar{\mathbf{W}}$: the reference word sequence

$$\frac{\partial O_{\text{with\_priors}}}{\partial u^t_k} = y^t_k - \frac{1}{p^\star(\bar{\mathbf{W}}|\mathbf{X})} \sum_{s \in lab(\bar{\mathbf{W}},k))} \alpha^\star_t(s)\beta^\star_t(s)$$

To understand the impact of label priors in the optimization process, we derive the gradients of the CTC loss with label priors. The star ($\star$) symbol to denote "with priors". E.g., $\alpha^\star_t(s)$, $\beta^\star_t(s)$ denotes the forward and backward probabilities with priors.

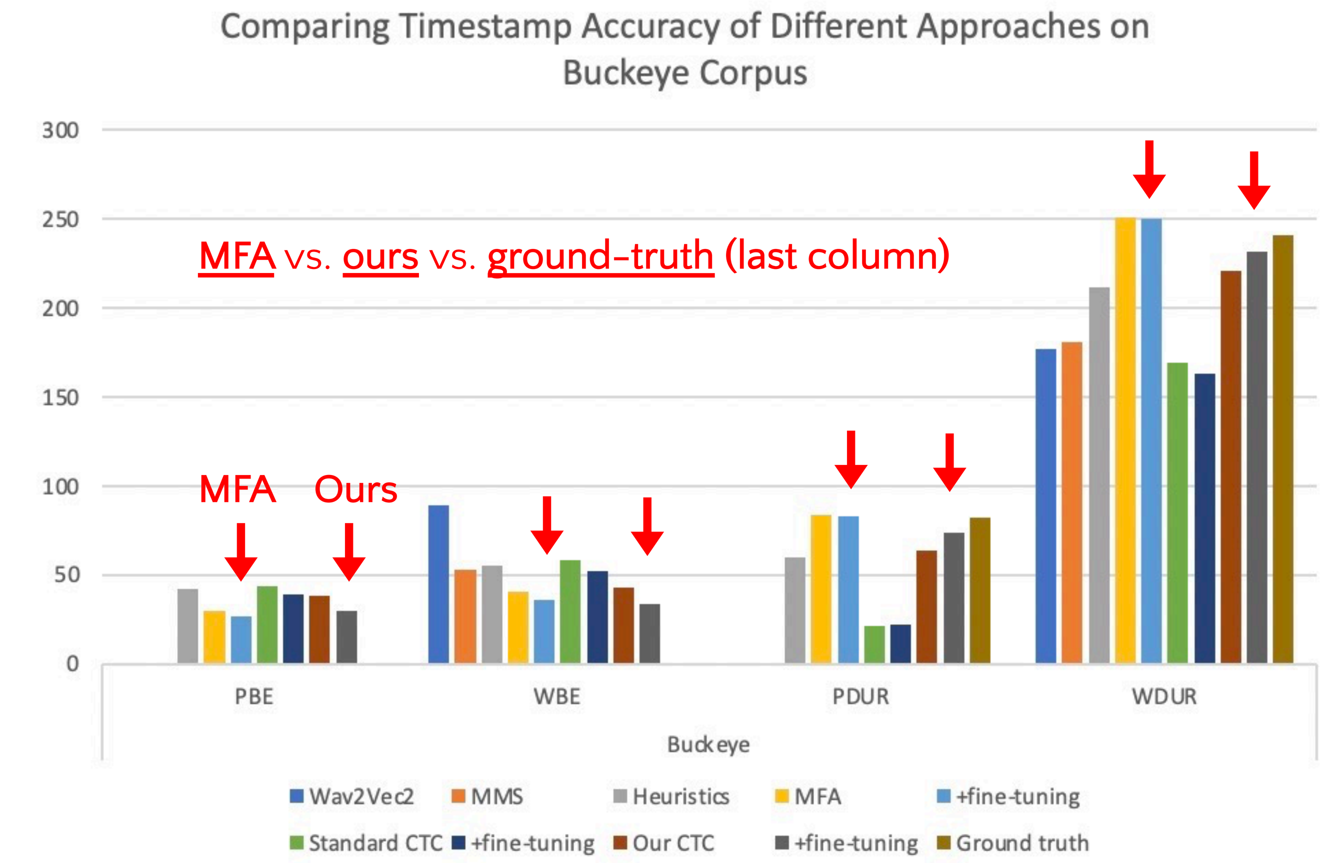
The optimization process just tries to match posterior $y^t_k$ with the proportion of $p^\star(W|X)$ going through the symbol k at time t after the label priors are applied. This is actually similar to standard CTC.

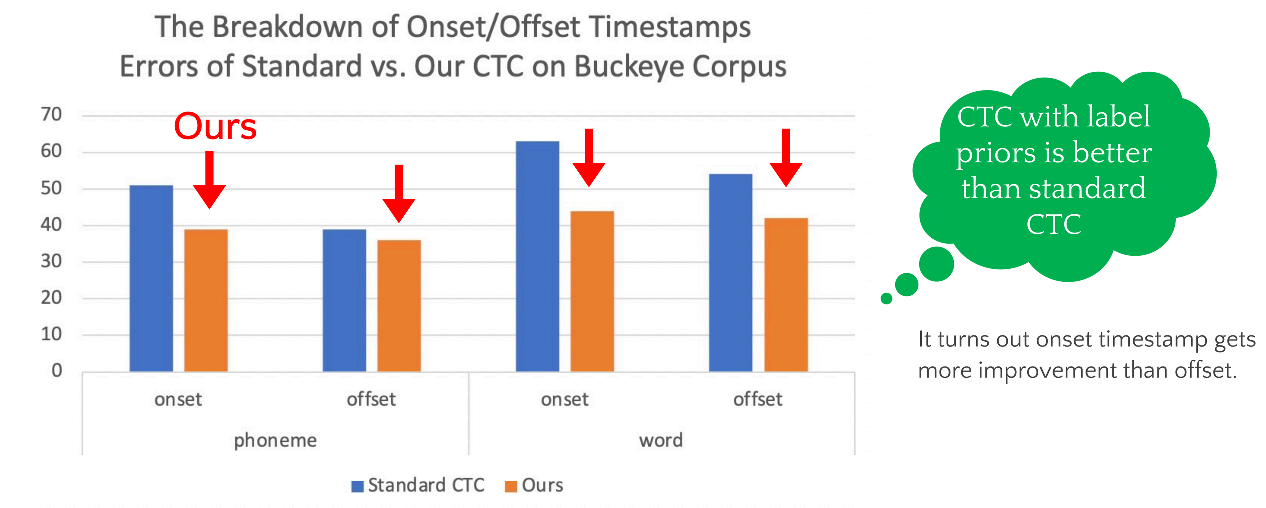
## Experiment Results

With a small TDNN-FFN model with **5M parameter**, our method **significantly** improves alignment accuracy over the **standard CTC model and a heuristic-based approach**. We also rival the state-of-the-art **Montreal forced aligner (MFA)** on **Buckeye and TIMIT corpus**, which contains human annotated timestamps. Nonetheless, our method has a simpler pipeline and faster runtime thanks to GPUs.

Metrics:

- Phoneme or word boundary error (PBE/WBE)
- Phoneme or word average duration (PDUR/WDUR)




Remarks:

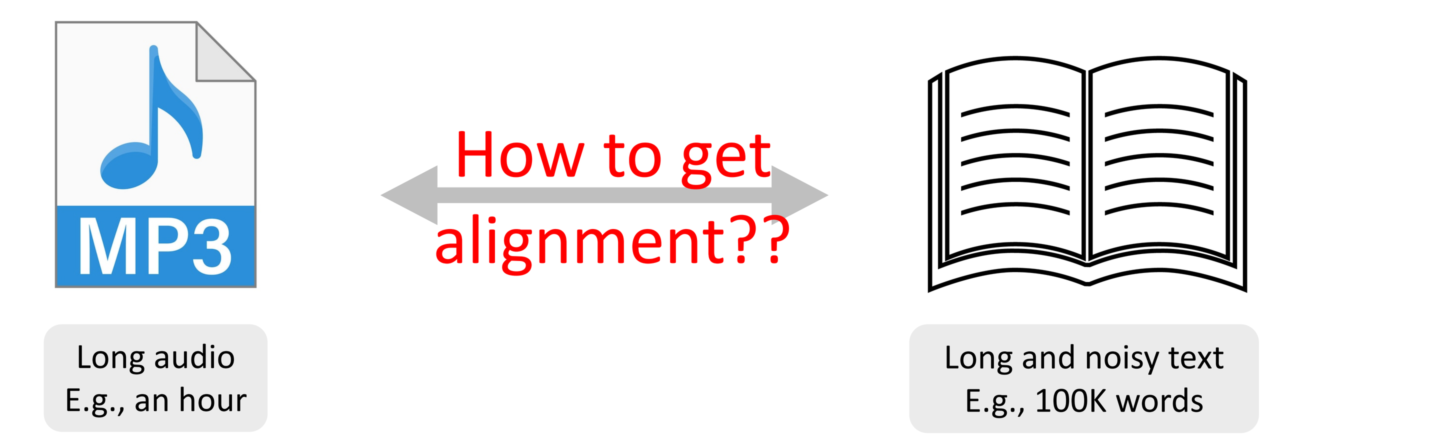
- We varied the neural net configurations (architecture, model size, modeling unit, and downsampling rate). It turns out TDNN-FFN phoneme model with 5M params and downsampling rate 2 has the best performance on Buckeye/TIMIT.
- Conformer or LSTM networks do not work well with label priors.
- From the experiments, applying label priors only during decoding, or applying penalties only to the blank tokens, or using an HMM topology does not have as good results as using the label priors. Some even degrades performance. These results agree with previous research.
- Fine-tuning the standard CTC model with the proposed loss works as good as training with our loss from scratch.

## One More Thing

Following the two popular TorchAudio forced alignment tutorials, we provide another tutorial on obtaining **accurate** speech-to-text alignment for long audio and noisy text.

In practice, we don't usually have small segments of audio and the corresponding exact and verbatim transcription to do forced alignment. We usually have the following:

- Long audio, which may not be suitable to be handled as a whole due to, e.g., limited CPU/GPU memory.
- Noisy long transcripts. It may have significant insertion, deletion or substitution errors.

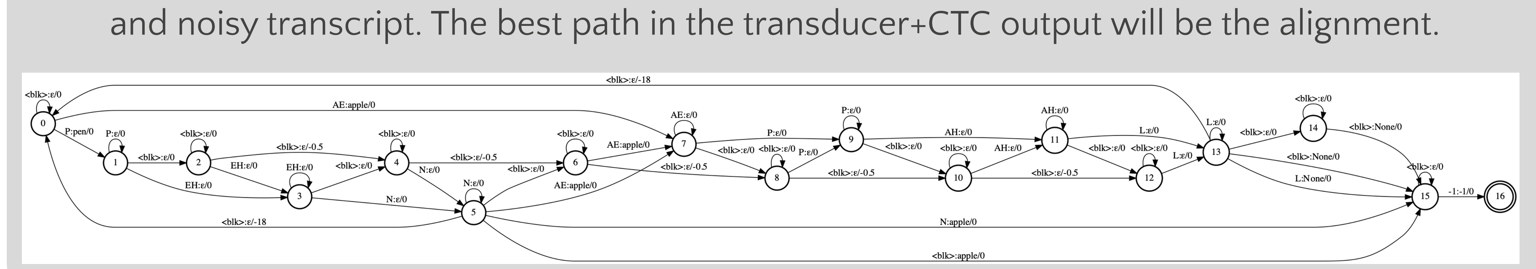



There are a few existing solutions:

- Kaldi (segment_long_utterances.sh), Gentle (https://github.com/lowerquality/gentle) and some other work employ a weighted finite state transducer (WFST) framework to model noisy texts. Gentle uses Kaldi's acoustic model and has an easy-to-use interface.
- WhisperX (https://github.com/m-bain/whisperX) uses attention mechanism to propose rough time stamps for uniformly segmented audio. Then, it performs phone-level or word-level forced alignment with an external aligner.
- MMS (https://arxiv.org/abs/2305.13516) uses a special *<star>* token to handle missing words.
- SailAlign iteratively identifies reliable regions and narrows down to align the remaining unaligned regions.

Our tutorial is based on WFST and thus falls in the first category. Our library and implementation is based on PyTorch. Any CTC model in PyTorch can be equipped with our library to become a robust aligner! This makes our aligner distinguish from existing ones.

We use a modified version of factor transducer to allow ins/del/sub errors in the long and noisy transcript. The best path in the transducer+CTC output will be the alignment.



We will demonstrate aligning a whole book, e.g., Walden by Henry David Thoreau (of 115K words), with its audiobook chapter (of 30 minutes) in the LibriVox project. This is similar to preparing the Librispeech corpus from raw data! However, today, we will have an easy-to-use, pretrained/finetuned neural network based solution! Please feel free to try it out! Scan the QR code to access the tutorial.