Align-Refine: Non-Autoregressive Speech Recognition via Iterative Realignment

Ethan A. Chi
Stanford University*
ethanchi@cs.stanford.edu

Julian Salazar
Amazon AWS AI
julsal@amazon.com

Katrin Kirchhoff
Amazon AWS AI
katrinki@amazon.com

Abstract

Non-autoregressive encoder-decoder models greatly improve decoding speed over autoregressive models, at the expense of generation quality. To mitigate this, iterative decoding models repeatedly infill or refine the proposal of a non-autoregressive model. However, editing at the level of output sequences limits model flexibility. We instead propose *iterative realignment*, which by refining latent alignments allows more flexible edits in fewer steps. Our model, Align-Refine, is an end-to-end Transformer which iteratively realigns connectionist temporal classification (CTC) alignments. On the WSJ dataset, Align-Refine matches an autoregressive baseline with a 14× decoding speedup; on LibriSpeech, we reach an LM-free test-other WER of 9.0% (19% relative improvement on comparable work) in three iterations. We release our code at https://github.com/amazon-research/align-refine.

1 Introduction

Transformer encoder-decoder models (Vaswani et al., 2017) have achieved high performance in sequence-to-sequence tasks like neural machine translation (NMT; Edunov et al., 2018) and end-to-end automatic speech recognition (ASR; Karita et al., 2019). However, like their recurrent predecessors, these models are *autoregressive* at inference: tokens are generated sequentially, with each token conditioned on all previous tokens. This makes decoding time linear in output sequence length, which is slow for long sequences. By contrast, non-autoregressive (NAR) models decode all output tokens independently and in parallel. When combined with self-attention, one gets fast, constant-time inference in NMT (Gu et al., 2018) and end-to-end ASR (Salazar et al., 2019). However, these models underperform their autoregressive counterparts, as the conditional independence between output tokens results in globally inconsistent outputs.[1]

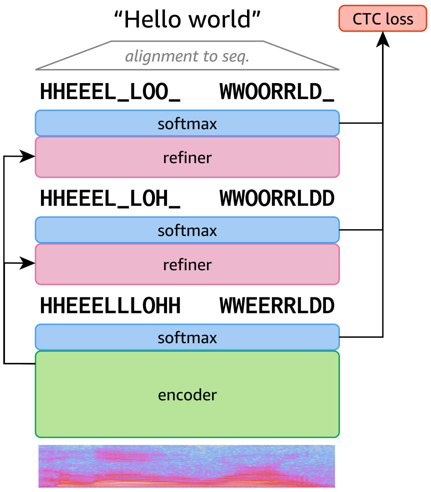

Figure 1: In Align-Refine, a Transformer encoder labels each input frame to give a latent alignment. The refiner, a non-causal Transformer decoder, improves the alignment conditioned on the encoder. After a bounded # of iterations, the result is collapsed into the output.

To mitigate these issues, *infilling* methods like Mask-Predict (Ghazvininejad et al., 2019) refine an initial non-autoregressive proposal, repeatedly predicting a *masked* subset of low-confidence proposal tokens in a fixed number of decoding passes. In ASR, most iterative non-autoregressive methods use infilling, like A-FMLM (Chen et al., 2020), Imputer (Chan et al., 2020), and Mask-CTC (Higuchi et al., 2020). However, during training, infilling requires partial proposals to be simulated by synthetically masking ground truths or samples from an expert. The resulting train-test mismatch leads to poor-quality generation. An alternative proposed in NMT is *iterative refinement* (Lee et al., 2018); here, full proposals are predicted and trained on at each iteration, with no masking required. This

* Work done during an internship at Amazon AWS AI.

[1] Gu et al. call this *the multimodality problem*, as it is induced by the highly multimodal distribution of target translations (or in the case of ASR, frame-level alignments).

reduces mismatch but still lacks flexibility: among other problems, working at the output sequence level constrains every iteration to the initial length L predicted by the model, making it difficult to correct insertions or deletions.

In this work, we propose *iterative realignment*, a variation on iterative refinement where *latent alignments* are edited instead. By working at the alignment level, we avoid length prediction and enable more powerful edits, while preserving the flexibility and reduced train-test mismatch of iterative refinement. Our model, Align-Refine, demonstrates this on ASR: after a Transformer encoder first produces a noisy non-autoregressive proposal, the *refiner* (a non-causal Transformer decoder) repeatedly conditions on both the previous proposal and the initial encoder representation to produce a better proposal. Both the encoder and refiner are supervised with CTC (Graves et al., 2006), a loss defined between sequences and their latent monotonic alignments. Unlike past methods, Align-Refine requires no token masking or expert policies.

We validate our approach on two English ASR benchmarks, improving on state-of-the-art infilling methods—Mask-CTC and Imputer—in word error rate (WER) and/or inference time (as measured by real-time factor[2], or RTF). On WSJ, we close the WER gap with an autoregressive baseline at 1/14th the RTF, outperforming Mask-CTC after a single iteration. On LibriSpeech, we improve on published (LM-free) NAR results by 2.1% WER absolute on test-other at <1/4th the effective layers (and thus estimated RTF) of Imputer. Our work suggests that iterative realignment is a promising direction for other sequence-to-sequence tasks, such as NMT.

2 Background

2.1 Connectionist Temporal Classification

CTC (Graves et al., 2006) is a strategy for defining latent monotonic *alignments* from an input sequence $\mathbf{x}$ to a shorter output sequence $\mathbf{y}$. Let '_', termed a *blank*, be an additional possible output token; then a *CTC alignment* is reduced to an output sequence by collapsing repeated labels then removing blanks, e.g., `AB__BB_A` ↦ `ABBA`. Since this is a many-to-one process, to calculate $p(\mathbf{y}|\mathbf{x})$, we marginalize over all alignments $\psi(\mathbf{y})$ mapping to an output $\mathbf{y}$. Assuming alignment labels are conditionally independent:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{a\in\psi(\mathbf{y})} p_\theta(a|\mathbf{x}) = \sum_{a\in\psi(\mathbf{y})} \prod_{t=1}^{|\mathbf{x}|} p_\theta(a_t|\mathbf{x}),$$

making $J_{\text{CTC}} = -\log p(\mathbf{y}|\mathbf{x})$ differentiable and efficiently computable via dynamic programming.

2.2 Existing Approaches

Iterative refinement methods non-autoregressively refine an initial *proposal* $\mathbf{y}^0$ of length L, with full re-predicted proposals $\mathbf{y}^k$ conditioned on previous proposals $\mathbf{y}^{k-1}$ and the input sequence $\mathbf{x}$:

$$p(\mathbf{y}^k|\mathbf{x}) = \prod_i p(y_i^k|\mathbf{x}, \mathbf{y}^{k-1}).$$

By contrast, *infilling methods* such as Mask-Predict (Ghazvininejad et al., 2019) *mask* part of the previous proposal every iteration. Only masked positions are re-predicted, conditioned on all unmasked tokens and the inputs $\mathbf{x}$:

$$p(\mathbf{y}^k_{\text{mask}}|\mathbf{x}) = \prod_i p(y_i^k|\mathbf{x}, \mathbf{y}^{k-1} \setminus \mathbf{y}^{k-1}_{\text{mask}}).$$

Typically, at each iteration a decreasing proportion of high-confidence tokens are unmasked[3] such that the full budget of K iterations is always required. Furthermore, state-of-the-art methods for English ASR—Mask-CTC (Higuchi et al., 2020, Figure 2) and Imputer (Chan et al., 2020)—enforce an added constraint that no decisions may be reversed at all: $p(y_i^k|\mathbf{x}) = \mathbf{1}[y_i^k = y_i^{k-1}]$ when $y_i^{k-1} \neq$ `<MASK>`.

3 Methods

We propose Align-Refine, a variation of iterative refinement which, like its predecessor, always keeps the proposal fully formed; this permits flexibility in decoding (as iteration can be stopped at any time) and potential speedups (errors seen and fixed in parallel; easy utterances refined in fewer steps).

However, unlike Lee et al. (2018), our proposals are latent CTC alignments $\mathbf{a}^k_{1:|\mathbf{x}|}$, not outputs $\mathbf{y}^k_{1:L}$. In previous work, working at the output sequence level is another source of irreversibility: the length L must be predicted even before the initial proposal $p_{\text{enc}}(\mathbf{y}^0|L, \mathbf{x})$ is generated, either explicitly (sampling from a modelled length distribution $p(L|\mathbf{x})$; Lee et al., 2018) or implicitly (collapsing CTC outputs before infilling; Higuchi et al., 2020). Either way, the decoder cannot fix insertions/deletions.

[2] The decoding time divided by the length of audio.

[3] In Mask-CTC, this is done by the decoder. In Imputer, full encoder passes are required, as there is no decoder.

```
                 Mask-CTC, 5 iterations                              Align-Refine (char.), ≤5 iterations

Enc  •••F••• ••S•••RE•••LY••EL•••• TO••••L• H• SAID  __SSA__FFFO_OD H'S DIRRECTLLY RRELATTED TOO HEALT HHE SAAIDD_
k=1  •••F••• H•S•D•RE••HLY••EL•••D TO•H••L• H• SAID  __SSA__ FFO_OD HIS DIRRECTLLY  RELATTED TO  HEALT  HE SAAID__
k=2  •••F••• HAS•DIRE••HLY•RELA••D TO•H•AL• H• SAID  __SSA__ FFO_OD HIS DIRRECTLLY  RELATTED TO  HEALTH HE SAAID__
k=3  S••F••• HAS DIRE•THLY•RELAT•D TO•H•ALT H• SAID  __SSAI_ FFO_OD HIS DIRRECTLLY  RELATTED TO  HEALTH HE SAAID__
k=4  S••FO•• HAS DIRE•THLY RELATED TO H•ALT HE SAID               (identical, so collapse early)
k=5  SE FOET HAS DIRECTHLY RELATED TO HEALT HE SAID                              ↓
End  SE FOET HAS DIRECTHLY RELATED TO HEALT HE SAID        SAI FOOD HIS DIRECTLY RELATED TO HEALTH HE SAID

                         Reference:  SEAFOOD IS DIRECTLY RELATED TO HEALTH HE SAID
```

Figure 2: WSJ dev93 utterance as decoded by our models. Mask-CTC's masks are denoted with '•'. Mask-CTC gives HEALT as it has no space for H from the very beginning, and outputs DIRECTHLY as it cannot undo HLY's emission at iteration $k = 1$. Meanwhile, Align-Refine makes the mistake HEALT immediately, but corrects it over two steps: at $k = 1$ it deletes HHE↦HE, and at $k = 2$ it sees the new space and inserts HEALT↦HEALTH.

By contrast, since CTC alignments are by nature fixed-length, Align-Refine easily handles insertions/deletions by placing/replacing blanks and spaces (Figure 2). At a given iteration k, we have:

$$p(\mathbf{y}|\mathbf{x}) = \mathbb{E}_{\mathbf{a}^{k-1}}\left[\sum_{\mathbf{a}^k \in \psi(\mathbf{y})} p_{\text{ref}}(\mathbf{a}^k|\mathbf{a}^{k-1}, \mathbf{x})\right].$$

This expectation over previous alignments is intractable, so like Lee et al. (2018) we take a deterministic lower bound: sampling the mode $\hat{\mathbf{a}}^{k-1}$. After we marginalize over all $\mathbf{a}^k$, the loss is $J_{\text{CTC}}(\hat{\mathbf{a}}^{k-1}, \mathbf{y}; \theta_{\text{ref}}, \theta_{\text{enc}})$. Since we don't know *a priori* which k is final, we apply the loss at $k = 1, \ldots, K$ with weights $w_1, \ldots, w_K$ for some hyperparameter K. For $k = 0$ we get $J_{\text{CTC}}(\mathbf{x}, \mathbf{y}; \theta_{\text{enc}})$.

In summary, we take the greedy alignment at each iteration and apply the CTC loss, as shown in Figure 1 for $K = 2$. In practice, we upweight the encoder and first iteration terms with weights λ and w_1, then sum to give the total loss. For this and other training details, consult Appendix B, C.

Data. We evaluate on two English ASR benchmarks: WSJ (81 hours; Paul and Baker, 1992) and LibriSpeech (960 hours; Panayotov et al., 2015). For WSJ, we run at the character level, matching Mask-CTC; for LibriSpeech, we build a 400-token BPE vocabulary, matching Imputer. We use 80-dim. filter banks and SpecAugment (Park et al., 2019).

Model. For WSJ we use a 12-layer encoder and 6-layer decoder, as in Higuchi et al. (2020); each layer has 4 heads over 256 units. For LibriSpeech, we use 8 heads over 512 units. With CNN frontends these are 27M and 71M parameters. Unless stated otherwise, we do $K = 4$ training iterations.

Decoding. We evaluate with decoding iterations k from $\{0, 1, 3, 5, 10\}$, exiting early on convergence (consecutive iterations are identical). The final CTC alignment is collapsed to give the result. To match previous non-autoregressive ASR work, we do not use a language model (LM).

4 Results

WSJ results (Table 1). Since Mask-CTC and Align-Refine share an identical architecture, the difference lies solely in training and evaluation. For both models, joint training with the refinement objective improves the encoder's performance as a standalone CTC model ($k = 0$) to a similar degree. However, from $k = 1$ onwards, Align-Refine outperforms Mask-CTC, improving the initial encoder proposal by 1.9% absolute in just one iteration:

Model	# passes		WER		RTF
	Enc	Dec	dev93	eval92	
Autoregressive baseline (Higuchi et al., 2020)					
CTC+ATTENTION	1	L	14.4	11.3	0.97*
+ beam search	1	$> L$	13.5	10.9	4.62*
Previous work (Higuchi et al., 2020; Chan et al., 2020)					
CTC	1	–	22.2	17.9	0.03*
MASK-CTC	1	0	16.3	12.9	0.03*
	1	1	15.7	12.5	0.04*
	1	5	15.5	12.2	0.05*
	1	#mask	15.4	12.1	0.13*
IMPUTER (8-LYR) †	8	–	–	12.7	–
Our work					
CTC	1	–	18.6	15.0	0.036
MASK-CTC	1	5	15.3	12.8	0.072
ALIGN-REFINE	1	0	16.2	13.5	0.037
	1	1	14.1	11.6	0.048
	1	3	13.9	11.5	0.066
	1	**5**	**13.7**	**11.4**	**0.068**
	1	10	13.7	11.4	0.068

Table 1: Non-autoregressive ASR on WSJ. No LMs are used. For CERs, see Table 4 (Appendix A). *: RTFs from Higuchi et al., which are lower than ours for corresponding models. †: No SpecAugment.

By $k = 5$, Align-Refine closes the performance gap with a comparable autoregressive model at 1/14th the RTF. As for the 8-layer Imputer, it seems un-

```
                         Align-Refine (subword), up to 5 iterations
Enc  [WHEN___[DI[DI_CKI__[CAME_____[DOWN________[HIS____[A__UNUNTT___[S__IGHT_LY__[S[SLA__PP_ T _[HIM
k=1  [WHEN___[DI[DI_CKIEE[CAME_____[DOWN________[HIS____[A__ _ UNTT___[SL_IGHT_LY__ _ [SLA__PP_ED_[HIM
k=2  [WHEN___[DI[DI_CKI_E[CAME_____[DOWN________[HIS____[A__ _ UN_T___[SL_IGHT_LY__ _ [SLA__PP_ED_[HIM
k=3                                    (identical, so collapse early)
...                                                  ↓
End                      [WHEN [DICKIE [CAME [DOWN [HIS [AUNT [SLIGHTLY [SLAPPED [HIM
```

Figure 3: LibriSpeech test-other utterance; reference matches the prediction. At $k = 1$ three separate corrections are made, two of which (DICKI↦DICKIE; SLAPPT↦SLAPPED) cannot be done from the audio. In $k = 2$ the multimodal prediction E E is resolved into _ E, though the repetition-collapsed transcript would be correct regardless.

likely that re-introducing SpecAugment would outperform this augmented 12+6-layer autoregressive baseline; even if performances did match, RTFs would be higher than Align-Refine's (Section 5).

LibriSpeech results (Table 2). Align-Refine gives 9.0% WER on test-other with no LM, outperforming published non-autoregressive models by 2.1% WER absolute. This is 5.1 points better than training the encoder with CTC only, even after SpecAugment is used (compare with 1.9 points from 16-layer CTC to Imputer). In Figure 3 we see the refiner make output-conditional edits that would be difficult for greedy CTC inference; we attribute our outsized gain on test-other to this LM-like behavior. Future work could start from stronger CTC encoders like QuartzNet (Kriman et al., 2020) to achieve even better results.

	# passes		**WER (test)**	
Model	**Enc**	**Dec**	**clean**	**other**
Autoregressive models (Han et al., 2020)				
LAS (360M)	1	*L*	2.6	6.0
RNN-T (CONTEXTNET-L)	1	*L*	2.1	4.6
Previous work (Chan et al., 2020; Kriman et al., 2020)				
CTC (16-LYR) †	1	–	4.6	13.0
IMPUTER (16-LYR) †	8	–	4.0	11.1
CTC (JASPER DR 10X5)	1	–	4.3	11.8
CTC (QUARTZNET 15X5)	1	–	3.9	11.2
Our work				
CTC	1	–	5.1	14.1
ALIGN-REFINE	1	0	4.6	11.5
	1	1	3.8	9.5
	1	**3**	**3.6**	**9.0**
	1	5	3.6	9.0

Table 2: Non-autoregressive ASR on LibriSpeech. No LMs are used. †: No SpecAugment.

5 Analysis

In all, we have seen that Align-Refine improves performance over infilling methods like Imputer and Mask-CTC across tokenizations and dataset sizes (Tables 1 and 2). We saw improvements qualitatively as parallel and multi-stage insertion/deletion edits (Figures 2 and 3). We conclude by discussing some properties, tradeoffs, and limitations:

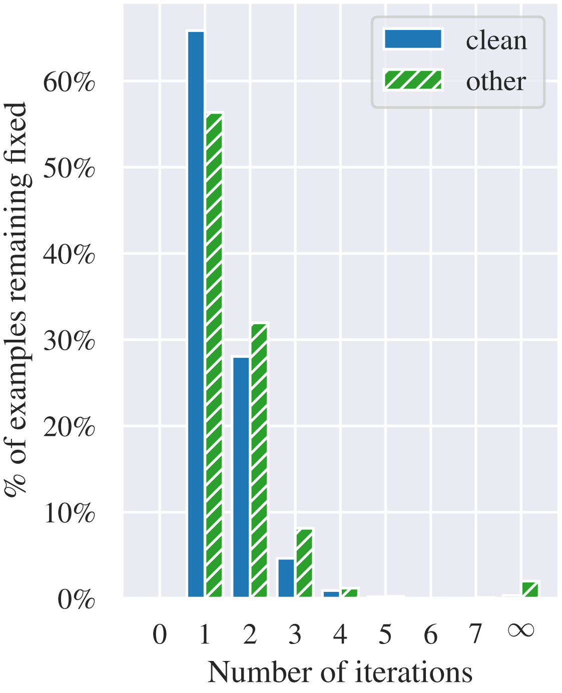

Figure 4: Proportion of utterances in each LibriSpeech test set whose alignments become fixed at iteration k. ∞ denotes utterances which cycle between alignments indefinitely.

Number of iterations. In Figure 4 we graph how many utterances became fixed at each iteration k (i.e., where $\hat{\mathrm{a}}^{k+1} = \hat{\mathrm{a}}^{k} \neq \hat{\mathrm{a}}^{k-1}$). First, we see that the CTC alignment is always revised by the refiner, evidence that CTC's non-autoregressive greedy mode is fundamentally different from the conditionally dependent mode annealed to by the refiner. We see that upweighting w_1 and training with $K = 4$ largely confined the "fixed point" iteration index k to 4 or less.

One interesting phenomenon is the small fraction of utterances which never reach a fixed point

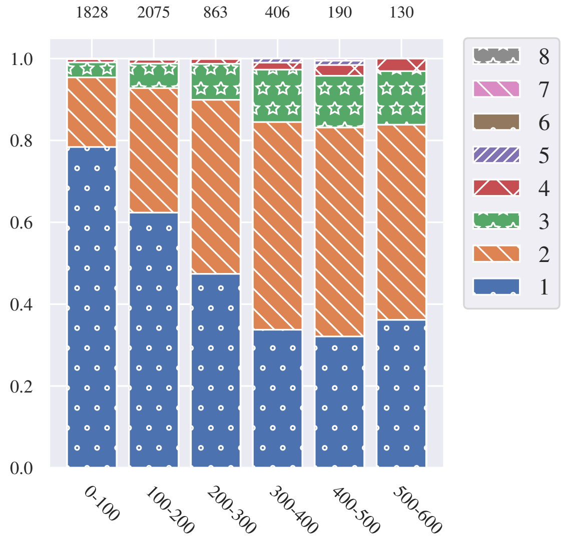

Figure 5: Bar plot of iterations at which each alignment became fixed, as a function of alignment length (4× downsampled from regular audio frames), over both `test-clean` and `test-other`. For comparability we normalize each bucket to 1.0; the total number of utterances in each bucket is above the chart.

(labeled ∞); rather, they cycle through two or more alignments repeatedly. In these cases, the transcript is finalized except on a local set of tokens where the model flips back and forth, e.g., `-OUS` versus `-US`. One could mitigate this by stopping after an edit distance of 1 between alignments is reached (similar to Lee et al., 2018), or by comparing with proposals e.g., two iterations prior.

Length independence. In Figure 5, we empirically validate the length independence of our method by plotting the "fixed point" iteration k versus the alignment length. The medians increase from $k = 1$ to $k = 2$ by lengths >300, but no further (for the lengths in LibriSpeech). One observation is that as an utterance gets longer, the chance that "multimodal" corrections like `[DI_CKI__`↦`[DI_CKIEE` (Figure 3) are made increases, which a further iteration then resolves (`[DI_CKIEE`↦`[DI_CKI_E`) independent of whether it affects the collapsed transcript. Interestingly, a few short alignments (e.g., lengths <100) use from 3 all the way up to 8 iterations, perhaps due to lack of linguistic context the refiner can use for disambiguation.

Speed and decoder depth. By factoring out refinement from feature processing, we can effectively adapt to variable inference budgets by adjusting the number of decoding iterations. While Imputer requires a separate run of the entire model for every iteration, for Align-Refine, only the decoder must be rerun. Another advantage is that since we have a full proposal at every iteration, in the vast majority of cases we exited early once the proposals stabilized.

Kasai et al. (2021) critiqued this factorization in non-autoregressive models by showing autoregressive models could shift layers from decoder to encoder to reduce the speed gap; however, we show that Align-Refine benefits from a similar reallocation (Table 3):

	WER (test-other) for each k				
Model-(# enc)-(# dec)	**0**	**1**	**2**	**3**	**5**
ALIGN-REFINE-12-6	12.5	10.0	9.4	9.3	9.3
ALIGN-REFINE-15-3	10.5	9.5	9.4	9.3	9.3
ALIGN-REFINE-17-1	10.4	10.0	10.0	10.0	10.0

Table 3: Results with various encoder-decoder splits on LibriSpeech. We take $K = 2$ to speed up training.

The refiner needs some depth for best performance: 17-1 underperforms on test-other (though within 0.1 points on test-clean). However, 15-3 performs as well as 12-6 at $k = 1$ onwards, despite passing through half the number of decoder layers. At $k = 3$, our LibriSpeech model's RTF was 0.171, while 15-3's RTF is 0.136. In this configuration, we pass through 15 encoder and at most $3 \times 3 = 9$ decoder layers. By contrast, Imputer inference passes through $8 \times 16 = 128$ encoder layers with the same alignment-length inputs and layer size.

Limitations. The refiner sometimes makes edits which do not affect post-collapse outputs (Figure 3, $k = 2$); variants that use repetition tokens (ASG; Collobert et al., 2016) or prohibit repetition collapse (Chan et al., 2020) may mitigate this behavior. The initial downsampling also restricts what edits can be done in one step (Figure 2, $k = 1, 2$).

We found that Align-Refine performed worse when there were more alignment labels per word (Appendix A). In general, word-level errors like `SAI FOOD HIS` (Figure 2) may require more coordinated mechanisms to fix. Future work could integrate non-autoregressive LMs like BERT (Devlin et al., 2019) via alignment synthesis, fusion, or otherwise (e.g., Salazar et al., 2020). Some concurrent works (Inaguma et al., 2021; Tian et al., 2021) also propose reranking NAR candidates with a jointly-trained autoregressive decoder.

Acknowledgements

We are grateful to the AWS Speech Science team and the Stanford NLP Group for their advice and feedback, to Yosuke Higuchi and William Chan for their helpful correspondence, and to the anonymous reviewers for their thoughtful feedback.

References

William Chan, Chitwan Saharia, Geoffrey E. Hinton, Mohammad Norouzi, and Navdeep Jaitly. 2020. Imputer: Sequence modelling via imputation and dynamic programming. In *ICML*.

Nanxin Chen, Shinji Watanabe, Jesús Villalba, and Najim Dehak. 2020. Listen and fill in the missing letters: Non-autoregressive transformer for speech recognition. *CoRR*, abs/1911.04908v2.

Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. 2016. Wav2letter: an end-to-end convnet-based speech recognition system. *CoRR*, abs/1609.03193v2.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-Transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *ICASSP*, pages 5884–5888. IEEE.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*, pages 489–500. Association for Computational Linguistics.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-Predict: Parallel decoding of conditional masked language models. In *EMNLP/IJCNLP*, pages 6111–6120. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *ICLR (Poster)*. OpenReview.net.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. 2020. ContextNet: Improving convolutional neural networks for automatic speech recognition with global context. In *INTERSPEECH*. ISCA.

Yosuke Higuchi, Shinji Watanabe, Nanxin Chen, Tetsuji Ogawa, and Tetsunori Kobayashi. 2020. Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict. In *INTERSPEECH*. ISCA.

Hirofumi Inaguma, Yosuke Higuchi, Kevin Duh, Tatsuya Kawahara, and Shinji Watanabe. 2021. Orthros: Non-autoregressive end-to-end speech translation with dual-decoder. In *ICASSP*. IEEE.

Shigeki Karita, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, Wangyou Zhang, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, and Ryuichi Yamamoto. 2019. A comparative study on transformer vs RNN in speech applications. In *ASRU*, pages 449–456. IEEE.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2021. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. In *ICLR*.

Samuel Kriman, Stanislav Beliaev, Boris Ginsburg, Jocelyn Huang, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, and Yang Zhang. 2020. QuartzNet: Deep automatic speech recognition with 1d time-channel separable convolutions. In *ICASSP*, pages 6124–6128. IEEE.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *EMNLP*, pages 1173–1182. Association for Computational Linguistics.

Toan Q. Nguyen and Julian Salazar. 2019. Transformers without tears: Improving the normalization of self-attention. In *IWSLT*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. LibriSpeech: An ASR corpus based on public domain audio books. In *ICASSP*, pages 5206–5210. IEEE.

Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. SpecAugment: A simple data augmentation method for automatic speech recognition. In *INTERSPEECH*, pages 2613–2617. ISCA.

Douglas B. Paul and Janet M. Baker. 1992. The design for the Wall Street Journal-based CSR corpus. In *ICSLP*. ISCA.

Julian Salazar, Katrin Kirchhoff, and Zhiheng Huang. 2019. Self-attention networks for connectionist temporal classification in speech recognition. In *ICASSP*, pages 7115–7119. IEEE.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *ACL*, pages 2699–2712. Association for Computational Linguistics.

Zhengkun Tian, Jiangyan Yi, Ye Bai, Jianhua Tao, Shuai Zhang, and Zhengqi Wen. 2021. One in a hundred: Select the best predicted sequence from numerous candidates for streaming speech recognition. *CoRR*, abs/2010.14791v3.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, pages 5998–6008.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. ESPnet: End-to-end speech processing toolkit. In *INTERSPEECH*, pages 2207–2211. ISCA.

A Additional Analysis

Character error rates (CERs). Given that the WSJ models operate on characters, we also report model CERs (where available) in Table 4, as a direct counterpart to Table 1:

	# passes	**dev93**		**eval92**	
Model	**Enc/Dec**	**CER**	**WER**	**CER**	**WER**
Autoregressive baseline (Higuchi et al., 2020)					
CTC+Attention ‡	1/L	5.5	14.4	4.0	11.3
+ beam search	1/>L	–	13.5	–	10.9
Previous work (Higuchi et al., 2020; Chan et al., 2020)					
CTC	1/–	–	22.2	–	17.9
Mask-CTC	1/0	–	16.3	–	12.9
	1/1	–	15.7	–	12.5
	1/5	–	15.5	–	12.2
	1/#msk.	–	15.4	–	12.1
Imputer (8-lyr) †	8/–	–	–	4.9†	12.7
Our work					
CTC	1/–	5.24	18.6	4.16	15.0
Mask-CTC	1/5	4.73	15.3	3.88	12.8
Align-Refine	1/0	4.75	16.2	3.92	13.5
	1/1	4.38	14.1	3.61	11.6
	1/3	4.34	13.9	3.61	11.5
	1/5	**4.33**	**13.7**	**3.60**	**11.4**
	1/10	4.33	13.7	3.60	11.4

Table 4: Non-autoregressive ASR on WSJ. No LMs are used. †: No SpecAugment, uses 400 BPE vocabulary. ‡: CERs retrieved from `https://github.com/espnet/espnet/blob/ffcf39c27f9aa0cf15c7ae7a06a8a9d35871602e/egs/wsj/asr1/RESULTS.md#cer-4`.

In Align-Refine, we see that an absolute reduction in CER per realignment leads to a 5x to 20x absolute reduction in WER, suggesting the corrected errors are largely character-level misspellings. Align-Refine also has smaller CERs for the comparable WER to the CTC+attention autoregressive model, e.g., Align-Refine has a CER of 4.4 for a WER of 14.1 while CTC+Attention has a CER of 5.5 for a WER of 14.4. This exhibits one difference in failure modes: when an autoregressive model predicts the wrong word, it is more likely to get the *entire word* wrong due to compounding error.

Speaking rate sensitivity. For some insight into how Align-Refine performance varies with approximate rate of speech, we split LibriSpeech test by the ratio of reference words to alignment length (4 × downsampled from regular audio frames). Our results are in Table 5:

words / align. lbl.	**test-clean**		**test-other**	
	words/utt.	WER	words/utt.	WER
[0.02, 0.09)	13.4	5.66%	10.1	14.10%
[0.09, 0.11)	21.2	3.60%	18.3	10.15%
[0.11, 0.13)	23.1	3.21%	20.9	7.58%
[0.13, 0.21)	20.0	2.99%	21.3	6.53%

Table 5: WER bucketed by approximate "speed" (as measured by words vs. alignment lengths) of both the test-clean and test-other evaluation sets. Each (bucket, set) pair contains 487 to 917 utterances.

Surprisingly, we find "slower" segments do worse (though this is partly confounded by low-word utterances having more silences). One possible explanation is that correcting a word requires coordinated change across multiple alignment labels, which is easier when there are fewer alignment labels per word. However, such "coordination" is hard as each position's edit is conditionally independent from other edits at any given iteration.

B Training Details

Our setup extends the Mask-CTC recipe (`https://github.com/espnet/espnet/pull/2223/`) in ESPnet (Watanabe et al., 2018). We release our Align-Refine recipe and ESPnet code changes at `https://github.com/amazon-research/align-refine`.

Architecture. Following Mask-CTC, the 12-layer encoder has a convolutional frontend that downsamples input lengths and features by 4×, using two layers of 2D convolutions of filter size 3×3 and stride 2 (Dong et al., 2018). The 6-layer decoder has no causal attention masks. Pre-norm residual paths are used (Nguyen and Salazar, 2019).

Training. The encoder and decoder could be trained separately (e.g. a refinement model that improves a pre-trained CTC model). In practice, we found that joint training was essential for the decoder to learn non-identity behavior (as it is trained on the initial encoder hypothesis and its own iterations, with no further noising). We apply dropout with $p = 0.1$ on WSJ and $p = 0.2$ on LibriSpeech. The initial CTC is weighted with $\lambda = 0.3$; remaining weights are spread over the K iterations, with w_1 three times larger than the rest. For WSJ, we used a batch size of 32 sequences. For LibriSpeech, we used a batch size of 25000 tokens. Gradients are accumulated in both cases. LibriSpeech models are trained over four V100 GPUs (p3.8xlarge instances on AWS). We trained to convergence on WSJ, and for 125K steps on LibriSpeech. Following Mask-CTC, we use label smoothing of 0.1 and the standard Transformer inverse square-root learning rate schedule with a linear warmup of 25000 steps. We double their `transformer-lr` factor to 10.0.

Decoding. At decode time, we average model weights over 30 training checkpoints. All RTFs are measured on a single CPU thread (`--nj 1`, `ngpu=0`). We take real-time duration (`time`) and divide by the total utterance duration in ESPnet's `utt2dur` file on `eval92` for WSJ and `test-other` for LibriSpeech.

Tuning. Hyperparameters were set by manual tuning, with hyperparameters chosen based on performance on the validation set.

C Dataset Details

We validate our method on two standard benchmarks in speech recognition:

WSJ (Paul and Baker, 1992) is a dataset of spoken English text derived from articles in the *Wall Street Journal* from 1987-1989. The training set, `SI-284`, consists of 81 hours of data; we validated on the `dev93` split (1.1 h) and evaluated on the `eval92` split (0.7 h). The dataset is downloadable at the following links: https://catalog.ldc.upenn.edu/LDC93S6A and https://catalog.ldc.upenn.edu/LDC94S13A.

LibriSpeech (Panayotov et al., 2015) is a dataset of spoken English text derived from audiobooks from the LibriVox project. The training set consists of 1000 hours of data. The validation and test sets are split by speaker into `clean` and `other` splits based on the performance of an acoustic model trained on WSJ. We validated on `dev-clean` (5.4 h) and evaluated on the `test-clean` (5.4 h) and `test-other` (5.1 h) splits. The dataset is downloadable at the following link: http://www.openslr.org/12/.