Interspeech 2022
18-22 September 2022, Incheon, Korea

Better Intermediates Improve CTC Inference

Tatsuya Komatsu[1], *Yusuke Fujita*[1], *Jaesong Lee*[2], *Lukas Lee*[2], *Shinji Watanabe*[3], *Yusuke Kida*[1]

[1]LINE Corporation, [2]NAVER Corporation,
[3]Carnegie Mellon University

komatsu.tatsuya@linecorp.com

Abstract

This paper proposes a method for improved CTC inference with searched intermediates and multi-pass conditioning. The paper first formulates self-conditioned CTC as a probabilistic model with an intermediate prediction as a latent representation and provides a tractable conditioning framework. We then propose two new conditioning methods based on the new formulation: (1) Searched intermediate conditioning that refines intermediate predictions with beam-search, (2) Multi-pass conditioning that uses predictions of previous inference for conditioning the next inference. These new approaches enable better conditioning than the original self-conditioned CTC during inference and improve the final performance. Experiments with the LibriSpeech dataset show relative 3%/12% performance improvement at the maximum in test clean/other sets compared to the original self-conditioned CTC.

Index Terms: connectionist temporal classification, self-conditioning, beam-search, language model

1. Introduction

End-to-end automatic speech recognition (ASR) directly maps input speech to text, realizing a simple ASR pipeline by eliminating the conventionally required hand-crafted pronunciation dictionary. Popular end-to-end ASR approaches are autoregressive models, such attention-based encoder-decoders [1] and recurrent neural network transducers [2], in which autoregressive decoders predict a token given previously estimated tokens with a sequence of encoded audio features. These autoregressive ASR methods have shown state-of-the-art performance, with powerful neural architectures such as Transformer [3, 4] and Conformer [5, 6]. Non-autoregressive ASR methods, as well, have been attracting interest because of their ability to perform fast inference that can predict all tokens simultaneously. Non-autoregressive ASR can be broadly divided into two types: methods based on non-autoregressive decoders that iteratively improves the decoded results [7, 8, 9, 10] and connectionist temporal classification (CTC) [11, 12] based methods. The CTC-based models seek to obtain a better representation by the audio encoder, using multi-task learning with hierarchical targets [13, 14, 15] and additional auxiliary losses based on additional modules [16]. In particular, interCTC and self-conditioned CTC [17, 18] have reported performance comparable to conventional autoregressive encoder-decoder models without language models in several benchmarks [19], although the decoding itself is a CTC greedy decoding (best path search).

InterCTC [17] and self-conditioned CTC [18] consist of a transformer/conformer-based audio encoder and a CTC decoder. The greedy decoding can be viewed as frame-by-frame classification of the text and has the disadvantage that the output text at each time step can only be predicted independently under conditional independence. A key feature of interCTC and self-conditioned CTC is how to capture textual information using only an audio encoder. InterCTC makes predictions of the output sequence in the intermediate layers and takes intermediate CTC loss as well as the final layer. Self-conditioned CTC extends interCTC to explicitly utilize intermediate predictions for conditioning the subsequent layers by adding the intermediate prediction to the input of the next layer. A recent comparative study [19] reported that self-conditioned CTC showed the best performance among the latest non-autoregressive models, without any special decoder for the output tokens. Thus, the conditioning framework using intermediate predictions has proven to be very powerful. However, the mechanism behind the conditioning has not been fully explained. Formulating this mechanism in a tractable form, such as a probabilistic model, allows for more advanced discussions and extensions.

In the sequence-to-sequence modeling, there are several works that formulate the probability model of the output as a marginalized distribution with a latent representation [20, 21, 22, 23]. In particular, recent research [24] has decomposed the entire task into its subtasks, and directly refining the latent representation of the subtasks to improve the overall performance. Self-conditioning, which explicitly handles intermediate prediction, could be similarly formulated as a probability model with intermediate prediction as a latent representation.

This paper reformulates the original self-conditioned CTC with an approximate probabilistic model. With this formulation, we introduce two other conditioning methods to inject external knowledge for improved inference: (1) conditioning using beam-searched intermediate outputs, (2) multi-path conditioning, in which the prediction results obtained from previous inference are used to condition the next inference. We confirm experimentally that the inference of the proposed method based on the new formulation yields further improvement on the strong self-conditioned CTC baseline.

2. Self-conditioned CTC

2.1. CTC-based ASR

Let $Y = (y_l \in \mathcal{V} \mid l = 1, \dots, L)$ be an L-length label sequence with vocabulary $\mathcal{V}$ and $X = (\mathbf{x}_t \in \mathbb{R}^D \mid t = 1, \dots, T)$ be a T-length audio feature sequence with feature dimension size D. End-to-end ASR can be formulated as a search problem of finding the most probable output sequence $\hat{Y}$ as follows:

$$\hat{Y} = \underset{Y}{\mathsf{argmax}}\, p(Y \mid X). \tag{1}$$

CTC models this $X \to Y$ mapping with a label probability for each time step and its alignment representing a text sequence.

Encoding of audio sequence: Let us consider a CTC-based system based on N-layer Conformer encoders. First, an audio sequence $X(= X^{(0)})$ is fed into the Conformer encoders and is converted to the N-th feature sequence $X^{(N)}$. Here, the input

10.21437/Interspeech.2022-11276

and output of each layer are represented as follows,

$$X^{(n)} = \mathsf{Encoder}^{(n)}(X^{(n-1)}). \tag{2}$$

Then, the vector $\mathbf{x}_t^{(N)}$ at time step t of $X^{(N)}$ is mapped to a probability distribution on the vocabulary labels and an extra blank symbol as $\mathcal{V}' = \mathcal{V} \cup \{\mathsf{blank}\}$, where the blank represents the repetition of the preceding token. This mapping is performed with a linear transformation and a softmax function as,

$$Z = \mathsf{Softmax}(\mathsf{Linear}_{D \to |\mathcal{V}'|}(X^{(N)})), \tag{3}$$

where $Z = (\mathbf{z}_t \in (0,1)^{|\mathcal{V}'|} \mid t = 1, \ldots, T)$ is a sequence of the label probability at each time step. The element $z_{t,k}$ is interpreted as the posterior probability of k-th label at time t. Note that we denote $\mathsf{Linear}(\cdot)$ and $\mathsf{Softmax}(\cdot)$ as operations to each vector in a sequence.

CTC decoding: Since Z is a sequence with the encoder input length, we need to align it to the text-domain sequence. CTC introduces an alignment path $A = (a_t \in \mathcal{V}' \mid t = 1, \ldots, T)$ and the collapsing function $\mathcal{B}(A)$ that removes all repeated labels and blank symbols in the alignment. The alignment a_t represents the label at time step t. a_t follows the obtained label distribution $a_t \sim \mathbf{z}_t$, and since label estimation is performed each time step, there is conditional independence $(a_t \perp\!\!\!\perp a_{\neq t} \mid X)$. Thus, the probability of alignment $p(A \mid X)$ is obtained as the product of the label probabilities over T time steps,

$$p(A = (k_1, ..., k_T) \mid X) = \prod_t p(a_t = k_t \mid X) \tag{4}$$

$$= \prod_t z_{t,k_t}. \tag{5}$$

Using this probability distribution over the alignment A, the probability $p(Y \mid X)$ in Eq. 1 is expressed as the sum of the probabilities of all possible alignment paths of Y, i.e.,

$$p(Y \mid X) = \sum_{A \in \mathcal{B}^{-1}(Y)} p(A \mid X). \tag{6}$$

To train neural network parameters using CTC, the loss is defined as the negative log-likelihood of Z for all possible paths corresponding to the target sequence $Y^{(\mathsf{tgt})}$,

$$\mathcal{L}_{\mathsf{ctc}}(Z, Y^{(\mathsf{tgt})}) = -\log \sum_{A \in \mathcal{B}^{-1}(Y^{(\mathsf{tgt})})} \prod_t p(a_t \mid X). \tag{7}$$

For inference, we need to find the most probable label sequence and its alignment based on the label probabilities at all time steps. However, this computational complexity grows exponentially with the sequence length, making it impractical. For the efficient search of the best alignment, prefix beam search [25] can be used with a language model. Joint CTC/attention decoding [26] is also a popular method using an attention-based decoder. When neither decoder nor language model is available, the best path search is a commonly used approximation. It assumes that the most probable label sequence Y^* and its alignment A^* corresponds to:

$$Y^* \approx \mathcal{B}(A^*) \tag{8}$$

$$\text{where } a_t^* = \arg\max_{a_t} p(a_t \mid X). \tag{9}$$

It is not guaranteed that the best label sequence will be found, but it is empirically confirmed that in most cases the solution is sufficient [19].

2.2. Conditioning with intermediate CTC

InterCTC [17] introduces additional CTC predictions from intermediate encoder blocks. An intermediate label prediction for the n-th encoder block $Z^{(n)} = (\mathbf{z}_t^{(n)} \in (0,1)^{|\mathcal{V}'|} | t = 1, \ldots, T)$ is computed similar to Eq. 3 as:

$$Z^{(n)} = \mathsf{Softmax}(\mathsf{Linear}_{D \to |\mathcal{V}'|}(X^{(n)})). \tag{10}$$

The losses for the intermediate predictions are computed as well as the original CTC loss (Eq. 7), and the total loss is a weighted sum of the original and the intermediate CTC losses:

$$\mathcal{L}_{\mathsf{ic}} = (1-\lambda)\mathcal{L}_{\mathsf{ctc}}(Z, Y) + \frac{\lambda}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} \mathcal{L}_{\mathsf{ctc}}(Z^{(n)}, Y), \tag{11}$$

where $\lambda \in (0,1)$ is a mixing weight and $\mathcal{N}$ is a set of layer indices for intermediate loss computation.

Self-conditioned CTC [18] further utilizes the intermediate prediction for conditioning the subsequent encoders. The intermediate prediction $Z^{(n)}$ in Eq. 10 is projected back to D-dimensional audio representation as

$$H^{(n)} = \mathsf{Linear}_{|\mathcal{V}'| \to D}(Z^{(n)}), \tag{12}$$

and added to the input of the next encoder:

$$X'^{(n)} = X^{(n)} + H^{(n)}, \tag{13}$$

where $n \in \mathcal{N}$ and $X'^{(n)}$ is a new input for the next encoder. $\mathsf{Linear}_{|\mathcal{V}'| \to D}(\cdot)$ maps a $|\mathcal{V}'|$-dimensional vector into a D-dimensional vector for each element in the input sequence.

3. Proposed method

In this section, we provide a probabilistic formulation for self-conditioned CTC to describe its behavior. We then propose a new formulation with best path approximation and introduce improved inference based on two new types of conditioning.

3.1. Probabilistic formulation of self-conditioning

Self-conditioned CTC achieves improved encoding by conditioning the audio encoder on intermediate predictions Y^{inter}. Eq. 1 can be formulated as a marginal distribution with latent representation as prior works [20, 21, 22, 23],

$$\hat{Y} = \operatorname*{\mathsf{argmax}}_Y \sum_{Y^{\mathsf{inter}}} p(Y, Y^{\mathsf{inter}} \mid X) \tag{14}$$

$$= \operatorname*{\mathsf{argmax}}_Y \sum_{Y^{\mathsf{inter}}} p(Y \mid Y^{\mathsf{inter}}, X) p(Y^{\mathsf{inter}} \mid X) \tag{15}$$

Self-conditioned CTC feeds hidden representation H^{inter} back to the encoder as Eq 13. H^{inter} is the linear projection of the posterior distribution Z^{inter}. Here, describing the linear transformation of Eq. 12 as $\mathbf{W} \in \mathbb{R}^{D \times |\mathcal{V}'|}$, the vector at each time step of H^{inter} can be written as follows,

$$\mathbf{h}_t^{\mathsf{inter}} = \mathbf{W}\mathbf{z}_t \tag{16}$$

$$= \sum_k \mathbf{w}_k z_{t,k}. \tag{17}$$

Let the vector $\mathbf{w}_k \in \mathbb{R}^d$ of $\mathbf{W}$ be the mapping $g(k)$ that maps the k-th label to the D-dimensional feature, and since $z_{t,k}$ is the

label probability distribution of the alignment $a_{t,k}$, Eq. (16) can be described as conditional expectation as follows:

$$\sum_k \mathbf{w}_k z_{t,k} = \sum_k g(a_{t,k}) p(a_{t,k} \mid X) \tag{18}$$

$$= \mathbf{E}\left[g(a_{t,k}) \mid X\right], \tag{19}$$

where $\mathbf{E}[\cdot]$ denotes the expectation. For an entire sequence, since $a_{t,k}$ at each time step is conditionally independent, we obtain

$$H^{\mathsf{inter}} = \mathbf{E}\left[g(A) \mid X\right] \tag{20}$$

$$= \mathbf{E}\left[g\left(\mathcal{B}^{-1}(Y)\right) \mid X\right]. \tag{21}$$

Thus, H^{inter} can be interpreted as the conditional expectation of Y given X. Therefore, self-conditioned CTC, which uses H^{inter} for conditioning, makes the following approximation of Eq. 14:

$$\sum_{Y^{\mathsf{inter}}} p(Y \mid Y^{\mathsf{inter}}, X) p(Y^{\mathsf{inter}} \mid X) \approx p(Y \mid \mathbf{E}[Y^{\mathsf{inter}}], X), \tag{22}$$

where the marginalization with Y is replaced by the conditioning on expectation with Y. This approximation depends on the 'confidence' of the probability $p(Y^{\mathsf{inter}} \mid X)$, and especially predictions at lower layers are often inadequate. This paper focuses and seeks better approximation of self-conditioning.

3.2. New formulation with best path approximation

This paper provides a new formulation of self-conditioned CTC by a different approximation to Eq. 14, that replaces the marginalization by maximization:

$$\hat{Y} = \underset{Y}{\mathsf{argmax}} \sum_{Y^{\mathsf{inter}}} p(Y \mid Y^{\mathsf{inter}}, X) p(Y^{\mathsf{inter}} \mid X)$$

$$\approx \underset{Y}{\mathsf{argmax}}\, p(Y \mid \tilde{Y}^{\mathsf{inter}}, X), \tag{23}$$

$$\text{where} \quad \tilde{Y}^{\mathsf{inter}} = \mathsf{argmax}_{Y^{\mathsf{inter}}}\, p(Y^{\mathsf{inter}} \mid X). \tag{24}$$

This method replaces the intractable marginalization by deterministically conditioning with the most probable single prediction, known as the Viterbi approximation.

To realize self-conditioned CTC by this approximation, Eq. 12 also requires an another form. First, we estimate the most probable label sequence $\tilde{Y}^{\mathsf{inter}}$ based on the intermediate label prediction, Z. As described in Section 2.1, we can use search techniques such as prefix beam search [25] with a language model, but we first formulate it with the best path approximation as in Eq. 8:

$$\tilde{Y}^{\mathsf{inter}} \approx \mathcal{B}(A^*) \tag{25}$$

$$\text{where} \quad a_t^* = \underset{a_t}{\mathsf{argmax}}\, p(a_t \mid X). \tag{26}$$

The resulting best path sequence A^* is transformed into a representation H by embedding it in D-dimensional feature:

$$H = \mathsf{Embedding}_{(1 \to D)}(A^*) \tag{27}$$

This operation Embedding projects the label of each time step a_t of A into the D-dimensional feature space, essentially the same as $g(\cdot)$ in Eq. 18. The resulting H is added to the next input as Eq. 13 to condition subsequent encoders.

The advantage of this approximation is that the conditioning used here is a deterministically obtained label sequence a_t^* in Eq. 26, rather than a sequence of probabilities or expectations $\mathbf{z}_t$, as introduced in Section 2.2. Therefore, to obtain a_t^* during inference, it makes it possible to apply sequence-level techniques like beam search, including combinations of external language models, to refine the information for conditioning $\tilde{Y}^{\mathsf{inter}}$ in Eq. 23. Accompanying this new approximation, we propose two new conditioning approach for inference.

3.3. Proposed inference with new conditioning approaches

3.3.1. Searched intermediate conditioning.

The best path approximation allows the text-domain sequence to be used for conditioning. Therefore, it is possible to obtain better intermediate prediction by using text-domain techniques, e.g., beam search with external language models:

$$Y^{\mathsf{search}} = \mathsf{Beam}\left(p\left(Y_l^{\mathsf{inter}} \mid X, Y_{1:l-1}^{\mathsf{inter}}\right)\right), \tag{28}$$

which is the prediction obtained by autoregressive LM, in contrast to the conditionally independent prediction in Eq. 25. But Y^{search} is the sequence of the text domain, which requires conversion to a time step alignment for conditioning. In this paper, we obtain the alignment of Y as the most probable path to Z by using Viterbi algorithm:

$$A^{\mathsf{search}} = \mathsf{Viterbi}(Y^{\mathsf{search}}, Z) \tag{29}$$

The resulting A^{search} is converted to the representation H for conditioning using Eq. 13.

3.3.2. Multi-pass conditioning

Another new conditioning method is multi-pass conditioning. This method is based on multiple inferences and uses the predicted results obtained in the previous inference as information for conditioning at the next inference. In each inference, the input features remain unchanged, and only the representation for conditioning H is replaced by the results obtained in the previous inference. Let $\hat{Y}^{(m)}$ denotes the final result obtained in the m-th inference, then $\hat{Y}^{(m)}$ is used in for condition in the $(m+1)$-th inference. At each conditioning layer, $\hat{Y}^{(m)}$ is aligned with intermediate prediction $Z^{(n)}$ and the obtained alignment is converted to $H^{(n)}$, using $\mathsf{Viterbi}(\cdot)$ and $\mathsf{Embedding}(\cdot)$ as Eq. 29. The result of the inference is naturally better than the prediction of the intermediate layers. Therefore, the condition based on that better result can be used to improve the next inference.

4. Experiment

To verify the effectiveness of the proposed method, we conducted experiments by using ESPnet [27, 6] with almost the same hyperparameters.

4.1. Data

The experiments are carried out using the LibriSpeech [28] dataset, utterances from read English audio books. We trained the models using the 100-hour subset (LS100) or the 960-hour full set (LS960), and we used the standard development and test sets for evaluation. For the input feature, 80-dimensional Mel-scale filterbank coefficients with three-dimensional pitch features were extracted using Kaldi toolkit [29]. Speed perturbation [30] and SpecAugment [31] were also applied to the training data.

Table 1: *Experimental results using greedy search and LM+beamsearch for the final outputs.*

model	output decoding	LibriSpeech (100h) dev-clean	dev-other	test-clean	test-other	LibriSpeech (960h) dev-clean	dev-other	test-clean	test-other
decoding with CTC greedy search									
CTC	greedy	10.21	24.19	10.65	25.00	3.79	9.76	3.93	9.88
InterCTC	greedy	7.76	20.59	8.16	21.21	3.09	8.49	3.30	8.36
SelfCond	greedy	7.51	20.04	7.86	20.65	2.86	7.87	3.06	7.91
SelfCond + best path	greedy	7.20	19.81	7.78	20.76	2.75	7.88	3.06	7.83
+ searched cond.[†]	greedy	**6.45**	**18.45**	**7.11**	**18.57**	**2.68**	**6.95**	**2.98**	**7.07**
+ 2-pass	greedy	6.95	19.25	7.51	20.23	2.71	7.23	3.01	7.32
decoding with Transformer LM + search									
SelfCond	beam search	5.05	15.44	5.40	15.61	2.19	5.96	2.45	6.08
SelfCond + best path	beam search	4.88	14.88	5.40	15.53	2.12	5.81	2.48	5.96
+ searched cond.[†]	beam search	**4.60**	**14.00**	**5.07**	**14.47**	2.17	5.66	2.48	5.97
+ 2-pass cond.	beam search	4.74	14.14	5.19	14.69	**2.07**	**5.53**	**2.44**	**5.77**

[†] For searched conditioning approache, we applied Transformer LM + beam search on the intermediate predictions.

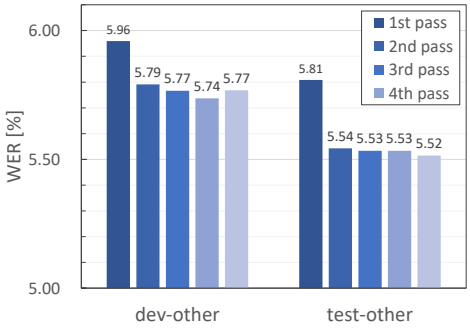

Figure 1: *Multi-pass results of LS960. The 1st pass is normal intermediate conditioning; 2nd and subsequent passes are conditioned on results of previous pass. The final output of each pass are obtained with beam search and the Transformer LM.*

Table 2: *Comparison of LMs used for searcher condition with LS960. Since the final output includes the performance of the LM+beam search, the results of the CTC greedy search are also shown for comparison of the encoder output.*

condition	decode	dev clean	dev other	test clean	test other
4-gram	beam	2.39	6.50	2.67	6.54
Transformer	beam	2.17	5.66	2.48	5.97
4-gram	greedy	2.77	7.51	3.02	7.29
Transformer	greedy	2.68	6.95	2.98	7.07
Oracle	greedy	2.24	5.18	2.49	5.42

4.2. Model configurations

CTC: We used the Conformer-CTC model as described in Section 2.1. The number of layers N was 18, and the encoder dimension D was 256. The convolution kernel size and the number of attention heads were 15 and 4, respectively. The feed-forward layer dimension in the Conformer blocks was set to 1024. The model was trained for 50 epochs, and the final model was obtained by averaging model parameters over 10-best checkpoints in terms of validation loss values. The batch-size is set to 1024 for LS960 and 128 for LS100.

InterCTC and +SelfCond: For intermediate CTC and self-conditioned CTC (Section 2.2), we applied five intermediate CTC predictions at the set of layer indices $\mathcal{N} = \{3, 6, 9, 12, 15\}$ with $\lambda = 0.5$ in Eq. 11, conditioning with the intermediate CTC predictions was applied according to Eq. 13. Other configurations are identical to the baseline CTC.

SelfCond + best path approx.: All configuration of the proposed model is identical with the SelfCond baseline except for the conditioning method as described in Section 3.2. For beam search of intermediate and final results, we used two different LMs: Transformer-based pretrained LM provided by ESPnet, and 4-gram LM trained with LibriSpeech 960h full set.

4.3. Results

Table 1 summarizes the experimental results. First, it can be seen that new self-conditioned CTC with the best path approximation boosts the performance of the strong self-conditioned CTC beseline. This can be explained because the conventional self-conditioned CTC uses expectation of the intermediate prediction that includes the ambiguity of the prediction. Of course, conditioning with deterministic predictions might have negative impacts if the predictions are wrong, but under this experimental settings, there were only positive effects.

Both conditioning by searched intermediates and multi-pass conditioning are shown to further improve the final results. In LS960, the proposed method shows superior results for all data sets, although the amount of improvement appears to be small because the results were already good before using the proposed conditioning. For the smaller LS100, the improvement by the proposed method is more significant.

Use of other language model Table 2 compares the results of using different LMs for searched conditioning (Section 3.3.1). It is clear that the more powerful LMs give better results. The greedy decoding of the final result can be seen as a refinement of embedding itself. It can be seen that performing conditioning with better information will also yield better final results. This table also shows the results of oracle conditioning, which is conditioning based on ground truth. This is the best performance that can be achieved with this architecture.

Multi-pass decoding Investigation of the impact of multi-pass conditioning (Section 3.3.2) are shown in Figure 1. The second pass shows the most WER improvement, and the third and later passes seem to be mostly saturated. The clean set shows the same trend as other set, although the improvement is slight because the 1st pass results are already high.

5. Conclusions

In this paper, we formulated self-conditioned CTC as a probabilistic model of output sequence marginalized with intermediate prediction for more tractable conditioning. On this formulation, we also introduced new conditioning approaches for improved inference; searched intermediate conditioning and multi-pass conditioning. Experimental results showed that our new formulation and conditioning improved the inference.

6. References

[1] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. NIPS*, 2015.

[2] A. Graves, "Sequence transduction with recurrent neural networks," in *ICML: Representation Learning Workshop*, 2012.

[3] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. ICASSP*, 2018, pp. 5884–5888.

[4] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang *et al.*, "A comparative study on transformer vs rnn in speech applications," in *Proc. ASRU*. IEEE, 2019, pp. 449–456.

[5] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[6] P. Guo, F. Boyer, X. Chang, T. Hayashi, Y. Higuchi, H. Inaguma, N. Kamo, C. Li, D. Garcia-Romero, J. Shi, J. Shi, S. Watanabe, K. Wei, W. Zhang, and Y. Zhang, "Recent developments on espnet toolkit boosted by conformer," in *Proc. ICASSP*, 2021, pp. 5874–5878.

[7] W. Chan, C. Saharia, G. Hinton, M. Norouzi, and N. Jaitly, "Imputer: Sequence modelling via imputation and dynamic programming," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1403–1413.

[8] N. Chen, S. Watanabe, J. Villalba, P. Żelasko, and N. Dehak, "Non-autoregressive transformer for speech recognition," *IEEE Signal Processing Letters*, vol. 28, pp. 121–125, 2021.

[9] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask CTC: Non-Autoregressive End-to-End ASR with CTC and Mask Predict," in *Proc. Interspeech*, 2020, pp. 3655–3659.

[10] E. A. Chi, J. Salazar, and K. Kirchhoff, "Align-refine: Non-autoregressive speech recognition via iterative realignment," in *Proc. NAACL*, 2021.

[11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. Association for Computing Machinery, 2006, pp. 369–376.

[12] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proc. ICML*. PMLR, 2016, pp. 173–182.

[13] S. Toshniwal, H. Tang, L. Lu, and K. Livescu, "Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition," in *Proc. Interspeech*, 2017, pp. 3532–3536.

[14] R. Sanabria and F. Metze, "Hierarchical multitask learning with ctc," in *Proc. SLT*, 2018, pp. 485–490.

[15] K. Krishna, S. Toshniwal, and K. Livescu, "Hierarchical multitask learning for ctc-based speech recognition," *arXiv preprint arXiv:1807.06234*, 2018.

[16] A. Tjandra, C. Liu, F. Zhang, X. Zhang, Y. Wang, G. Synnaeve, S. Nakamura, and G. Zweig, "Deja-vu: Double feature presentation and iterated loss in deep transformer networks," in *Proc. ICASSP*. IEEE, 2020, pp. 6899–6903.

[17] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *Proc. ICASSP*, 2021, pp. 6224–6228.

[18] J. Nozaki and T. Komatsu, "Relaxing the Conditional Independence Assumption of CTC-Based ASR by Conditioning on Intermediate Predictions," in *Proc. Interspeech*, 2021, pp. 3735–3739.

[19] Y. Higuchi, N. Chen, Y. Fujita, H. Inaguma, T. Komatsu, J. Lee, J. Nozaki, T. Wang, and S. Watanabe, "A comparative study on non-autoregressive modelings forspeech-to-text generation," in *Proc. ASRU*. IEEE, 2021.

[20] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *ICML*, 2018.

[21] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," in *Proc. ICLR*, 2018.

[22] J. Lee, E. Mansimov, and K. Cho, "Deterministic non-autoregressive neural sequence modeling by iterative refinement," in *EMNLP*, 2018.

[23] R. Shu, J. Lee, H. Nakayama, and K. Cho, "Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior," in *Proc. AAAI*, vol. 34, no. 05, 2020, pp. 8846–8853.

[24] S. Dalmia, B. Yan, V. Raunak, F. Metze, and S. Watanabe, "Searchable hidden intermediates for end-to-end models of decomposable sequence tasks," in *Proc. NAACL-HLT*, 2021.

[25] A. L. Maas, A. Y. Hannun, D. Jurafsky, and A. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *ArXiv*, vol. abs/1408.2873, 2014.

[26] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[27] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-End Speech Processing Toolkit," in *Proc. Interspeech*, 2018, pp. 2207–2211.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[29] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[30] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Proc. Interspeech*, 2015, pp. 3586–3589.

[31] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.