NON-AUTOREGRESSIVE END-TO-END AUTOMATIC SPEECH RECOGNITION INCORPORATING DOWNSTREAM NATURAL LANGUAGE PROCESSING

Motoi Omachi[1], *Yuya Fujita*[1], *Shinji Watanabe*[2], *Tianzi Wang*[3]

[1]Yahoo Japan Corporation, [2]Carnegie Mellon University, [3]Johns Hopkins University

ABSTRACT

We propose a fast and accurate end-to-end (E2E) model, which executes automatic speech recognition (ASR) and downstream natural language processing (NLP) simultaneously. The proposed approach predicts a single-aligned sequence of transcriptions and linguistic annotations such as part-of-speech (POS) tags and named entity (NE) tags from speech. We use non-autoregressive (NAR) decoding instead of autoregressive (AR) decoding to reduce execution time since NAR can output multiple tokens in parallel across time. We use the connectionist temporal classification (CTC) model with mask-predict, i.e., Mask-CTC, to predict the single-aligned sequence accurately. Mask-CTC improves performance by joint training of CTC and a conditioned masked language model and refining output tokens with low confidence conditioned on reliable output tokens and audio embeddings. The proposed method jointly performs the ASR and downstream NLP task, i.e., POS or NE tagging, in a NAR manner. Experiments using the Corpus of Spontaneous Japanese and Spoken Language Understanding Resource Package show that the proposed E2E model can predict transcriptions and linguistic annotations with consistently better performance than vanilla CTC using greedy decoding and 15–97x faster than Transformer-based AR model.

Index Terms— Speech recognition, natural language processing, linguistic annotation, end-to-end, non-autoregressive

1. INTRODUCTION

The end-to-end (E2E) model, which predicts an output sequence from an input feature sequence with a single neural network (NN), has been frequently used in automatic speech recognition (ASR) [1–7]. In ASR, the E2E model predicts grapheme/multi-graphemic sequences and generates transcriptions. However, in spoken language understanding, not only transcriptions but also linguistic annotations such as phonemes and part-of-speech (POS) tags are helpful [8].

The recently proposed E2E model predicts transcriptions and such linguistic annotations jointly [9–12]. For example, an E2E model [12] predicts transcriptions and phonemic sequences using one-to-many sequence mapping. However, it requires additional alignment post-processing to obtain the correspondence between phonemic sequence and grapheme sequences. As another example, E2E models [9–11] output a serialized sequence consisting of graphemic/multi-graphemic units followed by named entity (NE) tags, phonemes, or other linguistic annotations, as shown in Figure 1. This model does not require additional alignment post-processing. In these studies, the connectionist temporal classification (CTC) is frequently used. However, CTC is inadequate for predicting transcriptions and additional information due to strong conditional independent assumption. CTC ignores explicit relations among output tokens even though these tokens are related to each other.

We recently proposed a Transformer-based E2E model instead of the CTC model for predicting an aligned sequence of the graphemic unit, phonemes, and POS tags [13]. Transformer is reasonable for predicting sequences whose output tokens are related to each other because it does not use any conditional independence assumption. However, Transformer has room for improvement in execution time. Transformer requires at least N iterations to predict N-length output. Since the length of a serialized sequence comprising multiple types of tokens tends to be longer, Transformer's execution time increases. This paper aims to develop an E2E model predicting transcriptions and linguistic annotations simultaneously faster than Transformer with sufficient performance.

The E2E models used in ASR can be categorized into autoregressive (AR) [2–7] and non-autoregressive (NAR) [1, 14–19] models, and there are trade-offs between ASR performance and execution time. AR models, such as Transformer, achieve better ASR performance compared to NAR models at the expense of higher execution times. On the other hand, the NAR models, such as CTC using greedy decoding, predict the output sequence faster than the AR model thanks to the parallel computation across time. However, the NAR model performance, especially CTC performance, is worse than the AR model due to the lack of explicit output token dependency. To improve the performance, Mask-CTC applying the mask-predict [14, 20] to the CTC output is proposed [19]. Mask-CTC refines unreliable output tokens conditioned on reliable output tokens and audio embeddings using Decoder, which is not used for other NAR models [16–18].

This paper proposes using Mask-CTC [19] to simultaneously predict transcripts and linguistic annotations instead of using an AR model like Transformer. We expect Mask-CTC Decoder, i.e., conditional masked language model (CMLM), to capture explicit output token dependency. This is suitable for predicting sequences whose output tokens are related to each other, e.g., a single-aligned sequence of graphemes and linguistic annotations used in this study. This study is the first attempt to use the Mask-CTC for predicting sequences comprising multiple type tokens in a serialized format. We expect that the NAR model achieves faster prediction than Transformer-based AR model because it can predict multiple tokens in parallel across time. Also, the ASR performance is improved compared to vanilla CTC using greedy decoding since mask-predict refines CTC output errors by considering the conditional dependence. Since the target sequence comprises of multiple type tokens (e.g., graphemes, phonemes, and POS tags), we propose a type-wise mask-predict algorithm that iteratively updates one of the token types (e.g., graphemes only) for each iteration.

Experimental results showed our mask-pridict-based NAR model outperforms vanilla CTC using greedy decoding in ASR and is faster than Transformer using AR decoding. In addition, we found our model works adequately in downstream natural language processing (NLP) tasks, i.e., POS tagging and NE tagging.

2. JOINT MODELING OF E2E ASR AND LINGUISTIC ANNOTATION PREDICTION

2.1. E2E ASR

The objective of E2E ASR is to estimate the sequence of the output tokens $\mathbf{y} = \{y_m \in \mathcal{Y}\}_{m=1}^{L}$ from input feature sequences $\mathbf{X} = \{\mathbf{x}_i \in \Re^{D^{\text{in}}}\}_{i=1}^{I^{\text{in}}}$. Here, D^{in} and I^{in} denote the number of the dimension of the input feature and the length of the input sequence, respectively; and L and $\mathcal{Y}$ denote output sequence length and the token vocabulary. To predict the output tokens, a NN is trained to maximize the following objective function:

$$\mathcal{L} = \log p(\mathbf{y}|\mathbf{X}) = \sum_{m=1}^{M} \log p(y_m|y_1, \ldots, y_{m-1}, \mathbf{X}). \quad (1)$$

During run-time, the ASR predicts hypothesized tokens $\hat{\mathbf{y}}$ by $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}^*} \log p(\mathbf{y}|\mathbf{X})$, where $\mathcal{Y}^*$ denotes a set of hypotheses.

As one of the NNs which maximizes Eq. (1), Transformer [21] is used in the state-of-the-art E2E ASR. Transformer explicitly models the relationship among output tokens with this conditional AR modeling function.

2.2. E2E ASR predicting transcriptions and linguistic annotations

This study aims to predict a single-aligned sequence of words/graphemes and linguistic annotations such as phonemes, POS tags, and NE tags. Here, we define the number of sequences, including the graphemic subsequence $\mathbf{y}$ in Section 2.1 and additional linguistic annotation sequences, as K. To predict K sequences, NN is trained to maximize the following log-likelihood of the joint probability:

$$\begin{aligned}\mathcal{L} &= \log p(\mathbf{y}_1, \cdots, \mathbf{y}_K|\mathbf{X}) \\ &= \log p(\underbrace{y_{1,1}, \cdots, y_{M_1,1}}_{\text{The first sequence}}, \cdots, \underbrace{y_{1,K}, \cdots, y_{M_K,K}}_{K\text{-th sequence}}|\mathbf{X}), \end{aligned} \quad (2)$$

where $\mathbf{y}_k = \{y_{m,k} \in \mathcal{Y}_k\}_{m=1}^{M_k}$ denotes an M_k-length sequence of the k-th type of tokens or linguistic annotations; and $\mathcal{Y}_k$ denotes a set of the corresponding tokens.

To maximize Eq. (2), we proposed a Transformer-based model for predicting word/morpheme and the corresponding linguistic annotations simultaneously by regarding multiple sequences as a single sequence [13]. In our approach, each of the K output sequences is collapsed into a *single* series of S segments in the single sequence representation, where S equals the number of graphemes in the output sequence. For example, the k-th token type sequence is collapsed into S subsequences, i.e., $\mathbf{y}_k = (\bar{\mathbf{y}}_{1,k}, \cdots, \bar{\mathbf{y}}_{S,k})$ where $\bar{\mathbf{y}}_{i,k}$ denote subsequence of k-th token type sequence obtained by splitting $\mathbf{y}_k$ at word boundaries. Then, we obtain the i-th variable-length segment composed of aligned graphemic and linguistic annotation subsequences $\mathbf{s}_i = (\bar{\mathbf{y}}_{i,1}, \cdots, \bar{\mathbf{y}}_{i,K})$. To obtain $\mathbf{s}_i$, we use existing manual annotations to jointly align the *training* sets of the K output sequences types. Finally, we obtain a single-aligned sequence by expanding the aligned sequence of $(\mathbf{s}_1 \cdots, \mathbf{s}_S)$. For English data, we applied byte-pair encoding (BPE) [22] to the single-aligned sequence to reduce the sequence length. Fig. 1 depicts an example of the single-aligned sequence obtained using A Spoken Language Understanding Resource Package (SLURP) [23], which is used in this study. We used the single-aligned sequences as training targets for an AR prediction task with Transformer. In this way, Transformer implicitly learns to simultaneously predict and align K output sequences from an input feature sequence $\mathbf{X}$.

```
Graphemes:
 i  have  a   birthday  on   monday
NE tags:
[N] [N]  [N] [event]   [N]  [date]
Target sequence:
i [N] have [N] a [N] birthday [event] on [N] m on day [date]
```

Fig. 1. Example of the target sequence used in this study. Tokens in parentheses, such as "[date]" indicate NE tags, respectively.

Letting $y_i^\star \in \mathcal{Y}^\star$ where $\mathcal{Y}^\star = \bigcup_{k=1}^{K} \mathcal{Y}_k$ denote elements of the collapsed single-sequence representation, the joint log-likelihood (Eq. (2)) can be written as

$$\begin{aligned}\mathcal{L} &= \log p(\mathbf{y}_1, \ldots, \mathbf{y}_K|\mathbf{X}) \\ &= \sum_{m=1}^{M^\star} \log p(y_m^\star|y_1^\star, \ldots, y_{m-1}^\star, \mathbf{X}) \,. \end{aligned} \quad (3)$$

Note that this form is almost equivalent to the single sequence objective function in Eq. (1) except that the variable $y_m^\star$ takes values from the union of the K symbol sets that represent the K output sequences and the length of this sequence $M^\star = \sum_{k=1}^{K} M_k$, is the sum of the lengths of the K output sequences.

3. PROPOSED FRAMEWORK

3.1. Mask-CTC based joint modeling

There is room for improvement in execution time for Transformer-based model proposed in [13]. For example, it is not easy to parallelize this computation during inference since Transformer requires a left-to-right assumption. In terms of execution time, the CTC model is superior to Transformer-based AR model because the inference computation is efficiently parallelized across time. However, the CTC model is not suitable for predicting multiple types of tokens, which contextually depend on each other, e.g., graphemes and phonemes. This is because CTC model ignores explicit output token dependencies due to the conditional independent assumption. To solve this issue, in Mask-CTC [19], CTC model is jointly trained with CMLM, and the mask-predict [24] is applied to the CTC output sequence during run-time. Joint training of CTC model and CMLM improves both performances, and mask-predict refines the errors due to conditional independent assumption used in vanilla CTC.

In this study, we propose to use Mask-CTC instead of Transformer to predict the single-aligned sequence consisting of transcriptions and linguistic annotations jointly. Note that the target sequence of the proposed model includes multiple types of tokens (e.g., graphemes, phonemes, and POS tags) instead of a single type of tokens (e.g., graphemes only). Therefore, we consider two mask-predict methods. One is to apply mask-predict for all tokens in a collapsed sequence regardless of the token type. We call it global mask-predict. The other method predicts a masked token only in the restricted token type. This method is called type-wise mask-predict.

3.2. Global mask-predict

A global mask-predict refines all masked tokens without considering the token types, similar to [19]. We first predict the target sequence, including transcriptions and linguistic annotations shown in Fig. 1. We use the single-aligned sequence, as introduced in Eq. (3). For the simplicity, the following explanation omits * in Eq. (3). Here,

we define CTC alignment as $\hat{\mathbf{a}}^{\mathrm{ctc}} = \{a_i^{\mathrm{ctc}}\}_{i=1}^{I^{\mathrm{in}}}$. Note that the length of CTC alignment I^{in} is longer than the length of the target sequence $M^\star$ due to including token repetitions and extra blank symbols ϕ^{blank}. The blank symbols are inserted to adjust the output length to be the same length as the input. After the CTC alignment prediction, the token repetitions and the blank symbols are removed to obtain CTC's output sequence $\hat{\mathbf{y}}^{\mathrm{ctc}} = \{y_m^{\mathrm{ctc}}\}_{m=1}^{M^\star}$. Since CTC ignores the explicit dependence among the output tokens, some output tokens are not accurate. In Mask-CTC, such tokens are replaced with a masked token ϕ^{mask} as follows:

$$\hat{y}_m^{\mathrm{mask}} = \begin{cases} \phi^{\mathrm{mask}} & p(y_m^{\mathrm{ctc}}|\mathbf{X}) < p^{\mathrm{TH}} \\ \hat{y}_m^{\mathrm{ctc}} & \text{otherwise} \end{cases}, \tag{4}$$

where p^{TH} denotes a threshold of the probability to determine whether the output token is accurate or not. Similar to [25], Mask-CTC refines the masked tokens iteratively using the CMLM. In the n-th iteration, the masked tokens are updated based on the other tokens and input feature $\mathbf{X}$ as follows:

$$p_m^{(n)} = \max_w p(y_m = w|\hat{\mathbf{y}}^{(n-1)}, \mathbf{X}), \tag{5}$$

$$y_m^{(n)} = \arg\max_w p(y_m = w|\hat{\mathbf{y}}^{(n-1)}, \mathbf{X}), \tag{6}$$

$$\hat{y}_m^{(n)} = \begin{cases} \phi^{\mathrm{mask}} & p_m^{(\mathrm{n})} < p^{\mathrm{TH}} \\ y_m^{(n)} & \text{otherwise} \end{cases}, \tag{7}$$

where $\hat{\mathbf{y}}^{(n)} = \{\hat{y}_m^{(n)}\}_{m=1}^{M^\star}$ denote the updated sequence in the n-th iteration and $\hat{\mathbf{y}}^{(0)}$ is initialized by $\hat{y}_m^{\mathrm{mask}}$ in Eq. (4), i.e., $\hat{\mathbf{y}}^{(0)} = \{\hat{y}^{\mathrm{mask}}\}_{m=1}^{M^\star}$. The conditional probability in Eqs. (5) and (6) is computed using CMLM. In each iteration, not all of the masked tokens are updated, but $[M^{\mathrm{mask}}/N]$ tokens are randomly selected and updated, similar to [19], where $[\cdot]$, M^{mask} and N denote the round function, the total number of masked tokens and the number of iterations, respectively.

3.3. Type-wise mask-predict

We propose a novel mask-predict method for the sequence consisting of multiple types of tokens. Unlike the global mask-predict, the proposed mask-predict restricts the token types to update for each iteration.

In the n-th iteration, the masked tokens are refined depending on token type k. In each iteration, token type k is selected, and only the masked tokens on this selected type k are updated as follows:

$$k \leftarrow \mathrm{mod}(n, K), \tag{8}$$

$$\hat{y}_m^{(n)} = \begin{cases} \phi^{\mathrm{mask}} & p_m^{(n)} < p^{\mathrm{TH}(k)} \text{ and } \hat{y}_m^{(n)} \in \mathcal{Y}_k \\ y_m^{(n)} & \text{otherwise} \end{cases}, \tag{9}$$

where $\mathrm{mod}(\cdot)$ denote modulo operation,and $p_m^{(n)}$ and $y_m^{(n)}$ are computed using Eqs. (5) and (6), respectively. Eq. (9) means that only the masked tokens on selected type k are refined for each iteration unlike global mask-predict in Eq. (7). Unlike global Mask-CTC, we define a threshold of the probability depending on the type of token (denoted as $p^{\mathrm{TH}(k)}$ in Eq. (9)). The mask of the other token type ($k' \neq k$) is not updated in iteration n. Note that type-dependent token representation $y_{m,k}$ is only used for the above mask update. The conditional probability using CMLM is computed with the collapsed sequence representation based on Eqs. (5) and (6) so that the inter-type dependencies are explicitly considered.

4. EXPERIMENT

4.1. Setup

We investigated the following models, which predict a single aligned sequence of transcriptions and linguistic annotations, using the Corpus of Spontaneous Japanese (CSJ) [26] and SLURP [23]: self-attention-based CTC (**CTC**) [27], a hybrid Transformer trained with an auxiliary CTC objective (**Transformer**) [6] and Mask-CTC [19]. For Mask-CTC, we applied Global mask-predict or Type-wise mask-predict, which are referred to as **g-Mask-CTC** and **t-Mask-CTC** in the rest of this paper, respectively. For the linguistic annotations, we used phonemes and POS tags for CSJ and NE tags for SLURP. These annotations are extracted from the annotation labels of the corpus. Similar to our previous study [13], we reduce the vocabulary size on SLURP data to 250 by applying BPE. The implementation of the models is based on ESPnet [28]. During training, the CTC model was regularized with Transformer decoder in the multi-task learning fashion similar to [6]. Such regularization techniques yield a significant improvement over a vanilla CTC baseline [15]. We set the number of epochs during training to 300 for CTC/Transformer, 500 for Mask-CTC of CSJ, and 600 for Mask-CTC of SLURP. Mask-CTC requires more training epochs to converge compared with the AR models and CTC as discussed in [19]. We used greedy search algorithm for vanilla CTC decoding. For Transformer decoding, we considered the target sequence, including K types of tokens, as a single sequence and applied beam-search algorithm. We did not apply Transformer/CTC joint decoding [6] and the language model shallow fusion [29]. To control the execution time, we set the beam-size to [1,2,3,4,5] for Transformer decoding and the number of iterations to [0,5,10,20,30] for Mask-CTC decoding.

4.2. Evaluation criteria

We evaluate the performance on ASR, linguistic annotation prediction performance, and downstream NLP tasks: proper noun detection (PND) or disfluency detection (DD) on CSJ and NE prediction (NEP) on SLURP. The disfluency tags are defined as one of the POS tags, and detecting such a tag is one of the essential natural language understanding (NLU) tasks [30, 31]. For ASR performance, we computed the character error rate (CER), word error rate (WER), and real-time factor (RTF). CER and WER are measures of the quality of graphemic transcripts. RTF is a measure of the execution time of ASR, defined as the execution time for decoding divided by the time of the input feature sequence. For linguistic annotation prediction performance, we computed precision, recall, and F-score. For downstream NLP tasks performance, we computed Word-F1, Character-F1, and SLU-F1, which consider ASR error [23].

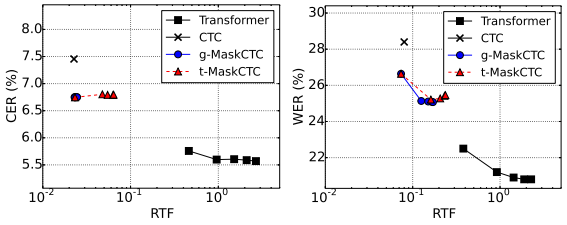
4.3. Speech recognition performance

Figure 3 shows speech recognition performance on CSJ[1] and SLURP. Since Transformer and Mask-CTC have a trade-off between execution time and performance, we controlled the hyperparameters, which affects the trade-off. We found that Transformer-based AR model achieves the best performance in CER and WER. However, RTF is higher compared to the vanilla CTC using greedy decoding and Mask-CTC. Mask-CTC shows quite a reasonable speed and performance trade-off, i.e., the performance is better than

[1]Due to space limitations, we showed the result of the eval3 only. We found the similar tendency for other evaluation dataset, i.e., eval1 and eval2.

```
CTC    : go [N] to [N] the [N] l i b r y  e   [N]   at [N] five [event] o ' clock [N]
=================================================================================================
ITR-0  : go [N] to [N] the [N] * i * * *  *    *    *  [N]  *      *     * * clock  *
ITR-2  : go [N] to [N] the [N] l i b * *  *    *    *  [N]  *      *     * * clock  *
ITR-3  : go [N] to [N] the [N] l i b * *  *    *    at [N]  *      *     * * clock  *
ITR-7  : go [N] to [N] the [N] l i b r ar y    *    at [N]  *      *     * * clock  *
ITR-8  : go [N] to [N] the [N] l i b r ar y [place] at [N]  *      *     * * clock  *
ITR-9  : go [N] to [N] the [N] l i b r ar y [place] at [N] five [time]  o ' clock [N]
=================================================================================================
REF    : go [N] to [N] the [N] l I b r ar y [place] at [N] five [time]  o ' clock [N]
```

Fig. 2. Prediction of the transcriptions and NE tags using Mask-CTC. ITR-n and REF denote the sequence refined by mask-predict of the n-th iteration and reference sequence, respectively; * denote the masked token.



(a) CSJ (b) SLURP

Fig. 3. Comparison among Transformer, vanilla CTC model, and Mask-CTC. **CER** and **WER** denote character error rate and phoneme error rate, respectively. **Transformer** is an AR model, and **CTC**, **g-Mask-CTC** and **t-Mask-CTC** are NAR models.

Table 1. Performance on the linguistic annotation prediction using CSJ. **Transformer**, **CTC**, **g-Mask-CTC**, and **t-Mask-CTC** predict graphemes and phonemes followed by POS tags from speech.

	System	Precision	Recall	F-score	RTF
Phoneme	[NAR] CTC	43.9	75.8	55.6	0.02
	[NAR]g-Mask-CTC	47.9	77.5	59.2	0.02
	[NAR]t-Mask-CTC	**49.3**	**78.0**	**60.4**	0.09
	[AR] Transformer	79.6	85.3	82.4	1.86
POS	[NAR] CTC	52.7	76.7	62.5	0.02
	[NAR] g-Mask-CTC	56.4	78.5	65.6	0.02
	[NAR] t-Mask-CTC	**57.0**	**78.8**	**66.2**	0.09
	[AR] Transformer	79.0	84.2	81.5	1.86

vanilla CTC thanks to the benefit of NAR modeling and still realize very fast inference compared with Transformer-based AR model.

Figure 2 depicts the NAR decoding example for utterance on SLURP development set. Unreliable tokens of the CTC output are replaced with the masked token, and these tokens are updated using unmasked tokens and audio embeddings during several iterations. Finally, we obtain an output token sequence similar to the reference.

4.4. Performance on Linguistic annotation prediction and downtream NLP tasks

Performance on linguistic annotations is listed in table 1. For Mask-CTC decoding, we tuned the threshold of the probability (p^{TH} in Eq. (7) and $p^{\mathrm{TH}(k)}$ in Eq. (9)) to minimize CER/WER on the development set with fixing the number of iterations to 50. We found that the Mask-CTC outperforms vanilla CTC consistently, and the proposed type-wise mask-predict yields better performance than the global mask-predict. Although Mask-CTC does not reach the score of Transformer, Mask-CTC is faster than Transformer [2].

[2]While we did not compare with the existing pipelined system, e.g., pipelined system of ASR followed by NLU, we expect our model to run faster and simpler than the pipelined system because our model does not require additional computation for NLU.

Table 2. Performance on downstream NLP tasks. **PND**, **DD**, and **NEP** denote proper noun detection, disfluency detection, and named entity prediction, respectively.

	System	Word-F1	Char-F1	SLU-F1	RTF
PND	[NAR] CTC	62.6	**63.8**	**63.2**	0.02
	[NAR]g-Mask-CTC	62.2	63.3	62.7	0.02
	[NAR]t-Mask-CTC	**62.7**	**63.8**	**63.2**	0.09
	[AR] Transformer	63.1	63.7	63.4	1.86
DD	[NAR] CTC	47.1	49.6	48.3	0.02
	[NAR] g-Mask-CTC	**47.6**	**49.9**	**48.7**	0.02
	[NAR] t-Mask-CTC	47.5	**49.9**	48.6	0.09
	[AR] Transformer	48.3	50.6	49.4	1.86
NEP	[NAR] CTC	62.1	69.6	65.6	0.07
	[NAR] g-Mask-CTC	**65.3**	**72.7**	**68.8**	0.15
	[NAR] t-Mask-CTC	**65.3**	72.5	68.7	0.16
	[AR] Transformer	72.5	76.3	74.3	2.27

The performances on downstream NLP tasks are listed in table 2. Note that the performance gaps between Transformer-based AR and NAR models are smaller on the PND and DD tasks than on the POS prediction task due to the less difficulty, i.e., PND/DD task predicts two labels while POS prediction task predicts 22 labels. Compared with Vanilla CTC, Mask-CTC yields better performance on DD and NEP tasks and comparable performance on the PND task. Although Mask-CTC with global mask-predict yields less performance than vanilla CTC on the PND task, the proposed type-wise mask-predict maintains the performance. We conclude that the proposed type-wise mask-predict is effective for Mask-CTC predicting sequences of transcription and linguistic annotations.

5. CONCLUSION

We proposed the NAR model to perform fast and accurate ASR and downstream NLP tasks simultaneously. Our model is based on Mask-CTC and predicts a single-aligned sequence including transcription and linguistic annotations. We investigated two types of mask-predict algorithm for Mask-CTC: global mask-predict and the proposed type-wise mask-predict. The differences among these algorithms are explicitly considering the type of the unreliable token, or not, for each iteration. The experiments on CSJ and SLURP showed that the proposed NAR model reduces the execution time significantly compared to Transformer-based AR model. We also found that the proposed Mask-CTC with type-wise mask-predict achieves better performance than the vanilla CTC on the downstream NLP tasks such as POS/NE tagging. In the future, we will improve the performance of our method and extend our method to streaming processing by block-wise processing similar to [32].

6. ACKNOWLEDGEMENTS

The authors would like to thank Mr. Yosuke Higuch of Waseda University for helpful discussions.

7. REFERENCES

[1] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, 2014, vol. II, pp. 1764–1772.

[2] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *in Proc. ICASSP*, 2013, pp. 6645–6649.

[3] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015, pp. 577–585.

[4] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[5] 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.

[6] S. Karita, N. E. Y. Soplin, S. Watanabe, M. Delcroix, A. Ogawa, and T. Nakatani, "Improving Transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration," in *Proc. Interspeech*, 2019, pp. 1408–1412.

[7] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *in Proc. ICASSP*, 2019, pp. 6381–6385.

[8] E. Simonnet, S. Ghannay, N. Camelin, Y. Estève, and R. D. Mori, "ASR error management for improving spoken language understanding," in *Proc. Interspeech*, 2017, pp. 3329–3333.

[9] S. Ghannay, A. Caubrière,, Y. Estève, N. Camelin, E. Simonnet, A. Laurent, and E. Morin, "End-to-end named entity and semantic concept extraction from speech," in *Proc. SLT*, 2018, pp. 692–699.

[10] K. Audhkhasi, B. Kingsbury, B. Ramabhadran, G. Saon, and M. Picheny, "Building competitive direct acoustics-to-word models for english conversational speech recognition," in *Proc. ICASSP*, 2018, pp. 4759–4763.

[11] H. Yadav, S. Ghosh, Y. Yu, and R. R. Shah, "End-to-End Named Entity Recognition from English Speech," in *Proc. Interspeech*, 2020, pp. 4268–4272.

[12] Y. Kubo and M. Bacchiani, "Joint phoneme-grapheme model for end-to-end speech recognition," in *Proc. ICASSP*, 2020, pp. 6119–6123.

[13] M. Omachi, Y. Fujita, S. Watanabe, and M. Wiesner, "End-to-end ASR to jointly predict transcriptions and linguistic annotations," in *Proc. NAACL*, 2021, pp. 1861–1871.

[14] N. Chen, S. Watanabe, J. Villalba, and N. Dehak, "Listen and fill in the missing letters: Non-autoregressive transformer for speech recognition," 2020.

[15] Y. Fujita, S. Watanabe, M. Omachi, and X. Chang, "Insertion-Based Modeling for End-to-End Automatic Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 3660–3664.

[16] W. Chan, C. Saharia, G. Hinton, M. Norouzi, and N. Jaitly, "Imputer: Sequence modelling via imputation and dynamic programming," in *Proc. ICML*, 2020, pp. 1403–1413.

[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*, 2021, pp. 3735–3739.

[19] 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*, 2020, pp. 3655–3659.

[20] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *Proc. EMNLP-IJCNLP*, 2019, pp. 6114–6123.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.-N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[22] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proc EMNLP*, 2018, pp. 66–71.

[23] E. Bastianelli, A. Vanzo, P. Swietojanski, and V. Rieser, "SLURP: A spoken language understanding resource package," in *Proc. EMNLP*, Nov. 2020, pp. 7252–7262.

[24] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Mask-predict: Parallel decoding of conditional masked language models," in *Proc. EMNLP*, Nov. 2019, pp. 6112–6121.

[25] Y. Goldberg and M. Elhadad, "An efficient algorithm for easy-first non-directional dependency parsing," in *Proc. NAACL*, June 2010, pp. 742–750.

[26] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of Japanese," in *Proc. of LREC*, 2000, pp. 946–952.

[27] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *Proc. Interspeech*, pp. 66–70, 2019.

[28] 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.

[29] T. Hori, J. Cho, and S. Watanabe, "End-to-end speech recognition with word-based RNN language models," in *Proc. SLT*, 2018, pp. 389–396.

[30] S. Wang, W. Che, and T. Liu, "A neural attention model for disfluency detection," in *Proc. of COLING*, Dec. 2016, pp. 278–287.

[31] P. Jamshid Lou, P. Anderson, and M. Johnson, "Disfluency detection using auto-correlational neural networks," in *Proc. of EMNLP*, Oct.-Nov. 2018, pp. 4610–4619.

[32] T. Wang, Y. Fujita, X. Chang, and S. Watanabe, "Streaming End-to-End ASR Based on Blockwise Non-Autoregressive Models," in *Proc. Interspeech 2021*, 2021, pp. 3755–3759.