A CTC Alignment-based Non-autoregressive Transformer for End-to-end Automatic Speech Recognition

Ruchao Fan, *Student Member, IEEE,* Wei Chu, Peng Chang, and Abeer Alwan, *Fellow, IEEE*

***Abstract*—Recently, end-to-end models have been widely used in automatic speech recognition (ASR) systems. Two of the most representative approaches are connectionist temporal classification (CTC) and attention-based encoder-decoder (AED) models. Autoregressive transformers, variants of AED, adopt an autoregressive mechanism for token generation and thus are relatively slow during inference. In this paper, we present a comprehensive study of a CTC Alignment-based Single-Step Non-Autoregressive Transformer (CASS-NAT) for end-to-end ASR. In CASS-NAT, word embeddings in the autoregressive transformer (AT) are substituted with token-level acoustic embeddings (TAE) that are extracted from encoder outputs with the acoustical boundary information offered by the CTC alignment. TAE can be obtained in parallel, resulting in a parallel generation of output tokens. During training, Viterbi-alignment is used for TAE generation, and multiple training strategies are further explored to improve the word error rate (WER) performance. During inference, an error-based alignment sampling method is investigated in depth to reduce the alignment mismatch in the training and testing processes. Experimental results show that the CASS-NAT has a WER that is close to AT on various ASR tasks, while providing a $\sim$24x inference speedup. With and without self-supervised learning, we achieve new state-of-the-art results for non-autoregressive models on several datasets. We also analyze the behavior of the CASS-NAT decoder to explain why it can perform similarly to AT. We find that TAEs have similar functionality to word embeddings for grammatical structures, which might indicate the possibility of learning some semantic information from TAEs without a language model.**

***Index Terms*—CTC alignment, non-autoregressive transformer, end-to-end ASR, intermediate loss.**

I. Introduction

End-to-end models have proven successful for speech recognition because of their ability to play the role of the acoustic, pronunciation, and language model in one single neural network [1], [2]. Training the above components together leads to fewer intermediate errors and thus a lower word error rate (WER) for ASR systems. This training mechanism also requires fewer model parameters, which is suitable for on-device deployment. Connectionist temporal classification (CTC) [3], attention-based encoder decoder (AED) [4], and RNN-Transducers [5], [6] are the most widely used end-to-end models. CTC has a high decoding efficiency when using the best path decoding strategy, but it is restricted by its assumption of conditionally independent outputs. AED, like the autoregressive transformer (AT) [7], [8], models output dependencies by incorporating a language-model-style decoder. However, the decoding in AT adopts an autoregressive mechanism for joint probability factorization, leading to a step-by-step generation of output tokens. Such a mechanism lowers the inference speed for ASR, which is an essential factor when designing an efficient ASR system.

Recently, non-autoregressive mechanisms have received increasing attention for their decoding efficiency, enabled by generating output tokens in parallel [9]–[14]. There are two major types of Non-Autoregressive methods for Transformers (NAT): (i) iterative NATs, and (ii) single-step NATs or one-shot NATs. The prevailing iterative NATs relax the strict non-autoregressive condition and iteratively generate outputs with K decoding passes. Thus, iterative NATs are sometimes called "semi-NAT". Single-step NATs, however, can generate the output sequence in one iteration. Different from the methods in neural machine translation that extend encoder input as the decoder input, single-step NATs for speech recognition extract high-level acoustic representations as the decoder input, assuming that language semantics can be captured by the acoustic representations [15]–[17]. These acoustic representations, however, are either implicit, extracted by attention mechanism [15] or incomplete using only CTC spikes [16], which make learning language semantics difficult.

Non-autoregressive methods continue to be proposed based on CTC because of its efficiency [18]. For example, Chi et al. proposed to train a refiner to iteratively improve CTC alignment based on the previous outputs of the refiner [19]. In [20], CTC alignment is enhanced with a mask token as a prior information for the decoder in each iteration. Nozaki et al. alleviate the output-independent problem of CTC by using intermediate predictions as additional inputs [21]. Furthermore, [22] improves the WER performance of a pure CTC model by a large margin using Wav2vec2.0 pretraining techniques. The performance of these methods, however, is still worse than their autoregressive transformer (AT) counterparts.

In this paper, we present a comprehensive study of a NAT framework by utilizing alignments over the CTC output space. The framework can generate the output sequence within one iteration, so we refer to it as CTC Alignment-based Single Step NAT (CASS-NAT). In CASS-NAT, there are four major modules: encoder, token-level acoustic embedding extractor (TAEE), self-attention decoder (SAD), and mixed-attention

R. Fan and A. Alwan are with the Department of Electrical and Computer Engineering, University of California, Los Angeles, CA, 90095 USA (e-mail: fanruchao@g.ucla.edu, alwan@ee.ucla.edu).

W. Chu and P. Chang are with PAII Inc., CA, USA, (email: {chuwei129,changpeng805}@pingan.com.cn).

This work is supported in part by the NSF and UCLA-Amazon Science Hub.

decoder (MAD). The encoder is used to extract a high-level acoustic representation for each frame. The TAEE extracts a more meaningful token-level acoustic embedding (TAE) using the information given by alignments over the CTC output space. The SAD and MAD model the dependencies between TAEs, where MAD considers encoder outputs directly for the purpose of source-attention while SAD does not. However, SAD indirectly uses the information from the encoder through the TAEs for self-attention. Since TAEs can be obtained in parallel, no recurrence in output sequence generation exists. Meanwhile, the two decoder modules can model the dependencies between TAEs in the latent space.

We summarize the contributions of this work as: 1) detailed experiments to examine the effect of various decoder structures on the WER, while the settings of SAD and MAD in [23] were intuitively selected; **2)** an investigation of the impact of the hyper-parameters on the proposed error-based sampling alignment (ESA) method, such as the sampling threshold, the number of sampled alignments, and the scoring model for ranking the sampled alignments. The investigation reveals a trade-off between accuracy and inference efficiency, which was not covered in [23]; **3)** comparisons of the effectiveness of each individual training strategy in [24] and their combinations are presented for a better understanding of the proposed training strategies. Knowledge distillation, which is not covered in [24], is also included in this work; **4)** an investigation of various encoder initialization schemes (including AT encoder, CTC encoder, and random initialization) for CASS-NAT training since the quality of the CTC alignment is highly relevant to model accuracy. The HuBERT encoder [25] is included in the comparison as well. Such an investigation was not done in earlier publications; and **5)** use the proposed methods on diverse datasets (LibriSpeech: adult-read, TED2: adult-spontaneous, MyST: child-spontaneous, and Aishell1: adult-Mandarin-read) to validate the generalizability of our algorithm, and obtain new state-of-the-art ASR results for non-autoregressive models. Our earlier publications reported results on only LibriSpeech and Aishell1.

The remainder of the paper is organized as follows. Section II introduces the background of end-to-end models and related work. Section III describes the CASS-NAT framework, including system architecture and training and inference strategies. Experimental setups are described in Section IV, and results are shown and discussed in Section V. We conclude the paper in Section VI.

II. Background

We first review the important concepts behind the proposed methods and provide basic notations. Let $X = (x_1, ..., x_t, ..., x_T)$ denote the input sequence, where x_t contains speech features of frame t. $Y = (y_1, ..., y_u, ..., y_U)$ is the output sequence, where y_u is a token at position u. The goal of speech recognition tasks is to find the best probable transcription Y given acoustic information X, which can be formulated as $Y^* = \underset{Y}{\operatorname{argmax}} P(Y|X)$.

A. Autoregressive Models

Recently, transformers are shown to be the best-performing autoregressive models (AT) [8], [26]. AT adopts an encoder-decoder structure, where the decoder generates each token conditioned on all previous tokens. This architecture design achieves sequence modelling by a chain of conditional probabilities, where each conditional probability constructs a classification problem. The AT model is then trained through an objective function as follows:

$$\begin{aligned} L_{AT} &= -\log P(Y|X) = -\log \prod_{i=1}^{U} P(y_i|y_{<i}, X) \\ &= -\sum_{i=1}^{U} \log P(y_i|y_{<i}, X) \end{aligned} \tag{1}$$

where P is the probability distribution of the AT model and $y_{<i}$ are all previous tokens before the i^{th} token.

AT can be trained efficiently by using all ground-truth tokens as the decoder input (teacher-forcing) or using parallel scheduled sampling [27]. During inference, however, a beam search algorithm is used to obtain the most probable sequences over the search space. The beam search and a requirement of using history tokens for generation, destroy the parallelism in AT, leading to low inference speed.

B. Non-autoregressive Models

Non-autoregressive models have no strict dependencies between tokens. For example, CTC has a conditional independence assumption, while Mask-CTC [14] trains the decoder as a masked language model to build a weak dependency between masked and unmasked tokens. By relaxing the dependency assumption, it is possible to generate all tokens in parallel, and thus increase inference speed.

1) CTC and its Alignment: The CTC model has no dependency assumption between tokens. The posterior probability of Y given X can be directly computed by a multiplication of the probability for each frame with each other. Let $Z = \{z_1, ..., z_t, ..., z_T\}$ be the output of the model, where z_t stands for the output at time step t corresponding to the input x_t. The length of Z, however, is always longer than that of Y for a speech recognition task. To compute the loss, a special blank token b is added in the vocabulary; b can be predicted as an output of Z at any time step. During inference, b and repeated tokens are removed to obtain a shorter sequence Y, where this process can be defined as a mapping rule β. During training, there exist multiple Zs that can be mapped to Y using the rule β. As a consequence, CTC loss is a summation of all such Zs, which is formulated as follows [3]:

$$\begin{aligned} L_{CTC} &= -\log P(Y|X) = -\log \sum_{Z \in \beta^{-1}(Y)} P(Z|X) \\ &= -\log \sum_{Z \in \beta^{-1}(Y)} \prod_{t=1}^{T} P(z_t|X) \end{aligned} \tag{2}$$

where β^{-1} is the inverse of the mapping rule.

An aligned relationship between all time steps in X and tokens in Y is presented in each Z. Thus, Z is also called an

CTC alignment, which is similar to the alignment in HMM-based ASR systems. In this paper, we consider the benefits of the information offered by the alignment Z to achieve a single-step NAT for end-to-end speech recognition. Z^* is referred to as the Viterbi-alignment when it has the maximum probability of being mapped to the ground truth Y.

2) Single-step NAT: Single-step NATs use the encoder-decoder structure, making the output the same length as Y. The output tokens are still conditionally independent of each other, just like CTC. But there is an implicit assumption that language semantics can be captured by high-level acoustic representations, which are similar to word embeddings. The model is updated using a cross entropy loss for each token, which can be formulated as:

$$L_{NAT} = -\log P(Y|X) = -\sum_{i=1}^{U} \log P(y_i|X) \tag{3}$$

C. Other Related Works

1) Convolution-augmented self-attention: The self-attention module in transformers captures global information by a weighted summation of the whole sequence. However, local information is also important for sequence modelling. Taking speech features as an example, frequency details in each vowel or consonant helps the recognition of these sounds. In computer vision, convolution layers have proved to be good at capturing local details within a kernel [28]. Recent works adopt this idea and augment transformers with a convolution module [29]–[31] in ASR. Relative positional encoding is also used in each self-attention module. The convolution-augmented self-attention block can effectively improve performance, but the improvement is only significant if applied in the AT encoder. The AT decoder, on the other hand, adopts a causal structure (upper triangular mask matrix for attention) that captures less local details with a convolution module, and thus the improvement would not be significant in that case. In this paper, we explore the use of convolution-augmented self-attention layers for the NAT decoder in addition to the encoder.

2) Intermediate Loss: Deep transformers always suffer from gradient vanishing, especially for parameters that are distant from the output layers. Intermediate loss has previously been proposed to add additional loss functions after each layer to boost the gradient update [32], [33]. It has been proven useful to use intermediate CTC loss for improving the performance of the CTC model [34], [35]. In [36], intermediate CE loss is used for training deep transformer-based acoustic models for an HMM-based hybrid ASR system. In this paper, we combine the usage of intermediate CTC and CE loss in the proposed framework.

3) Self-supervised Learning: Self-supervised learning (SSL) has been popular in recent years, especially for low-resource tasks [25], [37]. We use HuBERT in our experiments to further improve the system performance. During pretraining, the HuBERT model reconstructs the masked input given unmasked areas. We take advantage of self-supervised pretraining to improve the modelling capability of the encoder, and thus the quality of the CTC alignment, which is important for accurate language semantics modelling in the NAT decoder.

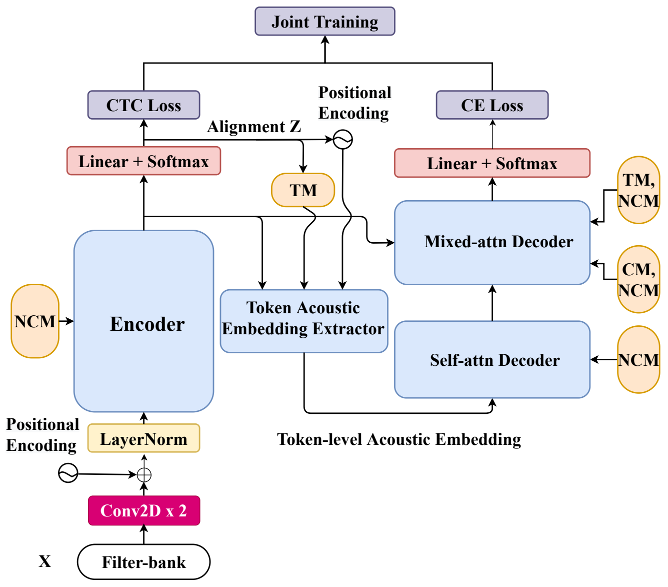


Fig. 1: An overview of the proposed CASS-NAT architecture. CM and NCM represent a causal and non-causal mask, respectively. TM stands for trigger mask.

III. Proposed Framework: CASS-NAT

In this section, we introduce the proposed CTC Alignment-based Single Step Non-autoregressive Transformer (CASS-NAT). We describe the mathematical derivation of the objective function, training strategies to improve performance, and the proposed sampling-based decoding strategy.

A. System Architecture

The proposed CASS-NAT system architecture builds upon the CTC/Attention hybrid architecture [38] to be non-autoregressive using CTC alignments. Fig.1 shows the four major modules in CASS-NAT: encoder, token acoustic embedding extractor (TAEE), self-attention decoder (SAD) and mixed-attention decoder (MAD).

1) ***Mask in attention****:* The attention mechanism is important for a transformer. The most basic computation in the attention mechanism is scaled dot-product self-attention using a sequence as input. Conventional self-attention utilizes information across the whole sequence. But each output of the self-attention mechanism is not necessarily dependent on all of the input sequence. For example, a neural language model uses an upper triangular matrix (mask matrix) to gather information from only past tokens in a sequence. In Fig. 1, we show three mask matrices for different purposes in the four major modules. Since we do not consider a streaming ASR model at this moment, non-causal mask (NCM) is used in the encoder. The NCM is a matrix where the paddings are zeros to prevent the padded tokens or padded frames from attention computation. In TAEE, a trigger mask (TM) is used to extract accurate token-level acoustic embeddings. TM marks out the triggered frames, such that the positions of used frames are marked as ones, while other positions are marked as zeros. Examples of TM can be found in Section III-A3. MAD contains both self-attention and cross-attention layers, which are similar to an AT decoder. In such cases, either a causal mask (CM) or NCM can be used for the self-attention layer, and either a NCM or TM can be used for the cross-attention layer. The different choices of the mask matrices in MAD are explored experimentally in Section V-A. Eventually, the self-attention computation can be augmented with a mask matrix as follows:

$$Attention(Q, K, V, M) = \left(Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \otimes M \right) V \tag{4}$$

where $Q \in R^{n_q \times d_q}$, $K \in R^{n_k \times d_k}$, $V \in R^{n_v \times d_v}$, and $M \in R^{n_q \times n_k}$ are the query, key, value and mask matrices, respectively.

*2) **Encoder:*** The encoder extracts high-level acoustic representations H from speech features X. A linear layer with a CTC loss function is added after the encoder as shown in Fig.1. The role of CTC is to obtain an alignment over the CTC output space to offer auxiliary information for the token acoustic embedding extractor (TAEE). During training, Viterbi-alignment is used. During inference, various methods for sampling from the CTC output space are explored experimentally.

*3) **Token Acoustic Embedding Extractor:*** The token acoustic embedding extractor is designed to extract token-level acoustic embedding (TAE) with the auxiliary information offerred by the CTC alignment. For example, given an alignment $Z = \{z_1, ..., z_t, ..., z_T\}$, we can estimate an acoustic segment for each token u as $\{t_{u-1}+1, ..., t_u\}$ (note that 1 here refers to one frame), and the number of tokens in Z.

First, CTC alignments offer an acoustic boundary for each token y_u, which is then transformed into the trigger mask. Specifically, we define a mapping rule from alignment to trigger mask, and fix the rule in both the training and inference phases. We regard the first non-blank index of each token in the alignment as its end boundary. The intuition is our assumption that the model will not output a token until it sees all the acoustic information.of the token. Using the first non-blank index is for simplicity and consistency in training and decoding. For example, if an alignment is $Z = \{\_, C, C, \_, A, \_, \_, T, \_\}$ for the ground truth $Y = \{C, A, T\}$, where _ is the blank symbol, the end boundary for C and A is Z_2 and Z_5, respectively, and thus the trigger mask for token A is $[0, 0, 1, 1, 1, 0, 0, 0, 0]$. The mapping rule might not be accurate for acoustic segmentations, but it should be consistent during the training and decoding. The trigger mask here is different from that used in [39] for streaming purposes. In [39], previous acoustic representations could be reused for each token, and thus the trigger mask in the previous example for token A is $[1, 1, 1, 1, 1, 0, 0, 0, 0]$.

Second, CTC alignments provide the number of tokens for the decoder input. After removing blank symbols and repetitions, the number of tokens in an alignment Z is used as the predicted length of sinusoidal positional encoding (decoder input length). As shown in Fig. 1, TAE for each token is then extracted with the trigger mask and the sinusoidal positional encoding using a one-layer source-attention block. The TAEs replace the word embeddings in AT to achieve parallel generation of each sequence.

*4) **Self-attention Decoder:*** TAEs extracted from TAEE (see Section III-A3) have the good property of parallel generation and thus is used as a substitution of word embeddings for the decoder input. Since there is no need to create recurrence in the decoder, we use a non-causal mask (NCM) in the self-attention decoder (SAD) to model the relationships between TAEs.

*5) **Mixed-attention Decoder:*** We assume that TAE has a similar capability of learning language semantics compared to the word embedding. Hence, we design a mixed-attention decoder (MAD) to retrace the encoder information for better decision making at the output layer. Similar to an AT decoder, MAD has a self-attention layer that uses either CM or NCM, and a source-attention layer that uses either TM or NCM. A linear layer is added after MAD, followed by a cross-entropy loss. Since we use the Viterbi-alignement during training, the output has the same length as the ground truth Y.

B. Training Details

The training criterion is presented in this section, followed by various training strategies used to improve the performance of CASS-NAT.

*1) **Training Criterion:*** In our framework, CTC alignments Z are introduced as latent variables. Given X and Y in Sec.II, the log-posterior probability can be decomposed into:

$$\log P(Y|X) = \log \mathbb{E}_{Z|X}[P(Y|Z, X)], \quad Z \in q. \tag{5}$$

where q is the set of alignments that can be mapped to Y. For those alignments that do not belong to q, we assume that $P(Y|Z, X)$ is equal to zero. To reduce computational cost, the maximum approximation [40] is applied:

$$\begin{aligned} \log P(Y|X) &\geq \mathbb{E}_{Z|X}[\log P(Y|Z, X)] \\ &\approx \max_Z \log \prod_{u=1}^{U} P(y_u | z_{t_{u-1}+1:t_u}, x_{1:T}) \end{aligned} \tag{6}$$

where $\mathbb{E}$ represents the expectation and t_u is the end boundary of token u ($t_0 = 0$). All t_us can be estimated from the alignment Z. We expect that TAEs can capture language semantics to a certain degree (see Section V-E for analysis), which helps alleviate performance degradation caused by the independence of the output tokens.

The framework is trained by jointly optimizing a CTC loss function on the encoder side (L_{CTC}) and a cross entropy (CE) loss function on the decoder side (L_{dec}) with a task ratio λ [38], and thus the final loss function (L_{joint}) is defined as:

$$L_{\text{joint}} = -\lambda \cdot \log \sum_{Z \in q} \prod_{i=1}^{T} P(z_i|X) - \log \prod_{u=1}^{U} P(y_u | z^*_{t_{u-1}+1:t_u}, X) \tag{7}$$

where P is the probability distribution, Z^* is the most probable alignment (Viterbi-alignment). The second term is a maximum approximation for the log-posterior probability as computed by Eq. 6.

*2) **Convolution Augmented Self-attention Block:*** The self-attention computation in Eq.4 considers global information across the sequence, but ignores local details. To alleviate this problem, convolution augmented self-attention blocks are proposed to emphasise the modelling of local dependencies of the input sequence in the encoder [30], [41]. Different from previous work, we apply the convolution augmented self-attention blocks in the SAD and MAD as well. Specifically, the feed-forward layer is decomposed into two sub-layers to be placed at the beginning and the end of the block. A convolution layer similar to that in [30] is inserted after the self-attention layer except that we empirically use layer normalization instead of batch normalization. The final computation in the i^{th} MAD can be formulated as:

$$\hat{s_i} = s_i + \frac{1}{2}\text{FFN}(s_i) \tag{8}$$

$$s_i^{'} = \hat{s_i} + \text{LN}(\text{Attn}(\hat{s_i}, \hat{s_i}, \hat{s_i}, \text{NCM})) \tag{9}$$

$$s_i^{''} = s_i^{'} + \text{Conv}(s_i^{'}) \tag{10}$$

$$s_i^{'''} = s_i^{''} + \text{LN}(\text{Attn}(s_i^{''}, H, H, \text{NCM})) \tag{11}$$

$$o_i = \text{LN}(s_i^{'''} + \frac{1}{2}\text{FFN}(s_i^{'''})) \tag{12}$$

where LN indicates layer normalization and FFN is the feed-forward network. NCM is non-causal mask. s_i and o_i are the input and output of block i, respectively. The convolution-augmented self-attention block can be used for other NAT models.

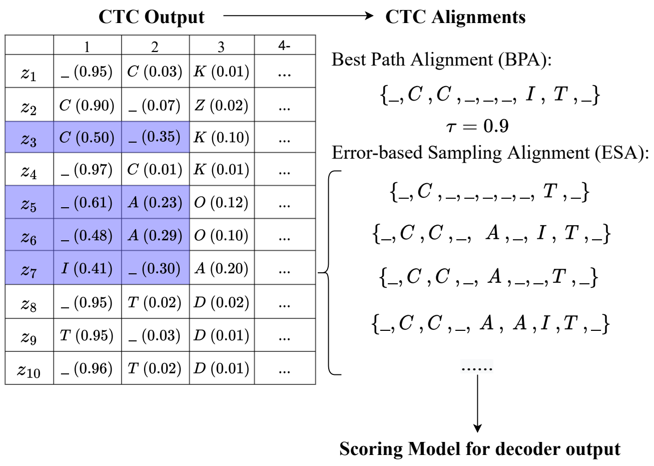
Different from the usage of relative positional encoding in [30], [42], we consider a maximum length of the relative position k as in [27]. Therefore, $2k+1$ position embedding are learned to represent the relative position between $[-k, k]$.

*3) **Intermediate Loss:*** Since CTC and CE loss functions are jointly optimized in the CASS-NAT framework, we incorporate intermediate CTC and intermediate CE loss functions into Eq.7 so that the parameters in different layers can be updated at the same scale. Let $L_{dec} = -\log\prod_{u=1}^{U} P(y_u|z^*_{t_{u-1}+1:t_u}, X)$ and $L_{CTC} = -\log\sum_{Z\in q}\prod_{i=1}^{T} P(z_i|X)$, the objective function is rewritten as:

$$\begin{aligned} L_{\text{joint}} = \lambda_{CE} L_{dec}^{final} + (1-\lambda_{CE}) L_{dec}^{mid} \\ + \lambda_{CTC} L_{CTC}^{final} + (1-\lambda_{CTC}) L_{CTC}^{mid} \end{aligned} \tag{13}$$

where λ_{CE} and λ_{CTC} are task ratios. mid and $final$ indicate the layer position of the inserted loss functions. We found intermediate loss to be more effective for CASS-NAT than AT models [24]. The intermediate loss can be added to other NAT models as well.

*4) **Trigger Mask Expansion:*** The quality of TAE relies on the accuracy of the trigger mask (TM), which is mapped from the CTC alignment. Although the CTC loss function is used to optimize the alignment, there are still errors when doing forced alignment over the CTC output space, leading to an inaccurate TM. To address this issue, we expand TM to include contextual frames for each token. For example, suppose the contextual frame size is one, the acoustic boundary of token U becomes $\{z_{t_{u-1}}, ..., z_{t_u+1}\}$. The trigger mask will then be expanded


CTC Output
CTC Alignments
Best Path Alignment (BPA):
$\{\_, C, C, \_, \_, \_, I, T, \_\}$
$\tau = 0.9$
Error-based Sampling Alignment (ESA):
$\{\_, C, \_, \_, \_, \_, \_, T, \_\}$
$\{\_, C, C, \_, A, \_, I, T, \_\}$
$\{\_, C, C, \_, A, \_, \_, T, \_\}$
$\{\_, C, C, \_, A, A, I, T, \_\}$
......
Scoring Model for decoder output

	1	2	3	4-
z_1	_ (0.95)	C (0.03)	K (0.01)	...
z_2	C (0.90)	_ (0.07)	Z (0.02)	...
z_3	C (0.50)	_ (0.35)	K (0.10)	...
z_4	_ (0.97)	C (0.01)	K (0.01)	...
z_5	_ (0.61)	A (0.23)	O (0.12)	...
z_6	_ (0.48)	A (0.29)	O (0.10)	...
z_7	I (0.41)	_ (0.30)	A (0.20)	...
z_8	_ (0.95)	T (0.02)	D (0.02)	...
z_9	T (0.95)	_ (0.03)	D (0.01)	...
z_{10}	_ (0.96)	T (0.02)	D (0.01)	...

Fig. 2: Illustration of obtaining CTC alignments from CTC outputs, including best path alignment (BPA) and error-based alignment sampling alignment (ESA). $C(0.90)$ indicates $P(z_i = C|X) = 0.90$. "_" stands for a blank token. The threshold τ for sampling is set to 0.9.

by one in the subsequent acoustic embedding extraction. Note that trigger mask expansion is designed specifically for CASS-NAT.

C. Inference: Error-based Sampling Decoding

During decoding, it is essential to obtain a CTC alignment that is close to the hypothetical Viterbi-alignment used in training. The transcription, however, is not available. We propose to use three different alignment generation methods for inference: (1) best path alignment (BPA); (2) beam search alignment (BSA); and (3) error-based sampling alignment (ESA). We also present the results of using Viterbi-alignment as an upper bound of WER, assuming that transcriptions are available during decoding, which is referred to as oracle alignment. **BPA** is similar to CTC greedy decoding that selects the token with the highest probability at each time step, but without removing blank and repetitive tokens in the final sequence. **BSA** is similar to beam search decoding over the CTC output space, which is the most probable alignment during decoding. Compared to BPA, BSA is supposed to generate an alignment that is closer to the oracle alignment, which could lead to a lower WER, but the parallelism of the CASS-NAT would be destroyed, resulting in a significant increase of the real time factor (RTF).

Considering the expectation in Eq. 6, we propose a third alignment generation approach by sampling based on the potential errors in BPA. The method is referred to as error-based sampling alignment (ESA). To generate ESA, a threshold τ determines whether sampling is required at each time step. If the highest probability of the output distribution is larger than τ, the rule of BPA holds. Otherwise, we randomly sample from the tokens with the first two largest probabilities. As shown in shaded blue in Fig. 2, CTC outputs at z_3, z_5, z_6, and z_7 need a sampling because of their low top-1 probability. According to the proposed sampling rule, four sampled alignments are shown as examples on the right side of Fig. 2. The reason of sampling within the top-2 tokens is that the trigger mask is sensitive to blank tokens and most mistaken outputs in BPA contain blanks in the top-2 tokens. In addition, sampling within 2 tokens is efficient because of the small sampling space. ESA aims at correcting the output which is prone to errors. Sampling in the decoding stage will not affect much inference speed because it can be done in parallel. It is possible that ESA-generated alignments are closer to the oracle alignment than BPA. Compared to BSA, ESA can be implemented in parallel, avoiding any increase in the RTF. Finally, either an AT baseline or language model can be used to score and identify the best overall alignment. Note that ESA can have different lengths for decoder input compared to BPA. As shown in Fig. 2, the decoder length is 4 for BPA (including an EOS token), while the lengths are 3, 5, 4, and 5 in the case of ESA. This fluctuation of the token numbers allows ESA to possibly sample an alignment that is of the same length as the oracle alignment, which is important for the performance of NAT models as will be shown experimentally. Note that ESA decoding is proposed specifically for CASS-NAT.

IV. Experimental Setup

A. Data Preparation

To examine the effectiveness of the proposed framework, we conduct several ASR tasks, including read and spontaneous speech, English and Mandarin speech, and adult and child speech. Four datasets are selected: (1) the 960-hour LibriSpeech English corpus [43] with read speech, (2) the 178-hour Aishell1 Mandarin corpus [44] with adult read speech, (3) the 210-hour TEDLIUM2 (TED2) English corpus [45] with TED talk speech, and (4) My Science Tutor (MyST) Kids English corpus [46] with spontaneous speech. We use the annotated part of MyST, which accounts for 42% (240 hours) of the corpus.

All experiments use 80-dim log-Mel filter-bank features, computed every 10ms with a 25ms Hamming window. Features of every 3 consecutive frames are concatenated to form a 240-dim feature vector as the input. The sets of output labels consist of 5k word pieces for LibriSpeech, and 500 word pieces for TED2 and for MyST. All sub-words are obtained by SentencePiece [47] using the training set of each dataset. 4230 Chinese characters are used as vocabulary for the Aishell1 dataset. To avoid overfitting, we applied speed perturbation [48] and SpecAugment [49] to the filter-bank features from TED2, Aishell1 and MyST. Speed perturbation is not used for LibriSpeech because of limited computational resources.

B. Network Architecture

1) Autoregressive Models: A CTC/Attention AT baseline is first trained with the architecture ($N_e = 12$, $N_d = 6$, $d_{ff} = 2048$, $nh = 8$, $d_{att} = 512$) for LibriSpeech, and ($N_e = 12$, $N_d = 6$, $d_{ff} = 2048$, $nh = 4$, $d_{att} = 256$) for the other three datasets, where d_{ff} is the dimension of the FFN module, d_{att} stands for the dimension of the attention module, nh is the number of attention heads, N_e and N_d are the number of encoder and decoder blocks, respectively. Prior to the encoder, two convolution layers with 64 filters, a kernel size of 3, and a stride of 2 is adopted, leading to a 4x frame-rate reduction. When using a conformer structure to the AT encoder, we reduce the d_{ff} to be 1024 to keep the number of parameters in the model the same as in the transformer baseline, and the maximum length of relative position k is set to 20. The kernel size in the convolution module is 31 for LibriSpeech and 15 for the other three datasets.

2) CASS-NAT: During training, CASS-NAT encoder is initialized with an AT encoder for faster convergence as in [50]. The decoder in AT baseline is replaced by 1-block TAEE, m-block SAD and n-block MAD. m and n are investigated using the LibriSpeech dataset, and the best setting is applied to the other three datasets. For convolution-augmented decoder, the dimension of feed-forward layers is also halved and the maximum length of relative position is set to 8 for the tasks with word piece units and 4 for the Aishell1 data. The contextual frame of trigger mask expansion is set to 1 because we do not see further improvements with a larger expansion. The intermediate loss functions are inserted in the middle layer of the encoder and MAD with λ_{CE} of 0.99 and λ_{CTC} of 0.5. The inserted projection layers are discarded during inference. These settings were chosen empirically.

C. Training and Decoding Setup

All experiments are implemented with Pytorch [51] [1]. The Double schedule in [49] is adopted as the learning schedule for the LibriSpeech and Aishell datasets, where the learning rate is ramped up to and held at 0.001, then be exponentially decayed to 1e-5. The transformer-based scheduler in [7] with warm-up steps of 25k and noam factor of 5 is used for the TED2 and MyST datasets. Layer normalization, dropout with rate of 0.1 and label smoothing with a penalty of 0.1 are all applied as the common strategies for training a transformer. We compute WERs of development sets for early stopping (no improvement for 11 epochs). The last 12 epochs are averaged for final evaluation. Most experiments end within 90 epochs. In order to investigate the impact of different models for scoring alignments in ESA, a transformer-based language model is trained with the provided text in LibriSpeech. The provided n-gram models in the dataset are also compared in the experiments. In terms of the pretrained HuBERT encoders, we use the Fairseq model that is trained from LibriSpeech 960-hour corpus for finetuning the English data model and the Tecent model that is trained from 10000-hour WenetSpeech [52] for finetuning the Mandarin data model.

During AT decoding, the beam size is set to 20 for LibriSpeech, and 10 for the other three datasets. No external language models are used during beam search decoding. The evaluation of the real time factor (RTF) is conducted using a V100 GPU with a batch size of one. For CASS-NAT decoding with ESA, the threshold, number of sampled alignments and scoring models are investigated in Section V-B.

V. Results and Analyses

The first set of ASR experiments used the LibriSpeech dataset to explore various decoder structures, impact factors

[1]Our code will be available at https://github.com/Diamondfan/cassnat_asr

TABLE I: WERs for different block numbers of the self-attention decoder (SAD) and mixed-attention decoder (MAD) using LibriSpeech. Various mask matrices used in MAD are also explored, where CM, NCM and TM represent causal mask, non-causal mask and trigger mask, respectively. For example, NCM + TM indicates that NCM is used for self-attention in MAD and TM is used for source-attention in MAD. Bold numbers represent the best results.

Decoder Structure	Mask in MAD	dev-clean	dev-other	test-clean	test-other
7SAD + 0MAD	-	5.3	11.1	5.4	11.1
5SAD + 2MAD	NCM + NCM	**4.7**	**10.4**	**4.8**	**10.3**
	NCM + TM	4.7	10.5	4.9	10.5
	CM + NCM	4.8	10.7	4.9	10.6
3SAD + 4MAD	NCM + NCM	4.7	10.4	4.8	10.4
1SAD + 6MAD	NCM + NCM	4.6	10.5	4.7	10.4

in ESA decoding, CTC alignment behaviour, and the effectiveness of the proposed training strategies. Using the best settings found with the LibriSpeech task, experiments are run with the other three datasets.

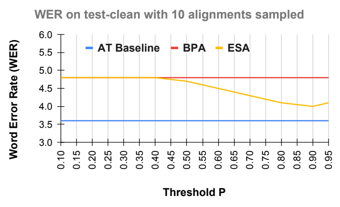
A. The Structure of CASS-NAT Decoder

To investigate the structure of the CASS-NAT decoder, we use various combinations of m SAD blocks and n MAD blocks ($m+n = 7$) in the decoder to keep the number of model parameters similar to that of the AT baseline. We use best path alignment during inference and discuss other decoding strategies in the next section. Results are shown in Table I. As shown in the table, not using MAD (0 MAD in the first row) causes a big performance degradation compared to other configurations, indicating that token-level speech representations might have to retrace the fine-grained frame-level information (encoder outputs) for better contextual modelling. The best performance is achieved when using 5 SADs and 2 MADs (WER of 10.3% on the test-other set).

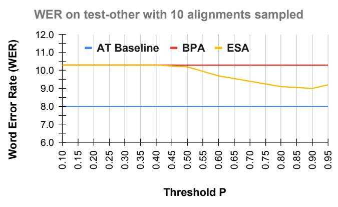
Because there are two attention layers (self-attention and source-attention) in each MAD, we try different mask matrices for the two layers. For the self-attention layer, either a causal mask (CM) or non-causal mask (NCM) is considered. For the source-attention layer, either a trigger mask (TM) or NCM is used. The results in Table I show that using NCM on both attention layers results in the best WER. Although TAE is regarded as a substitution of word embedding, CM seems not necessarily required in the CASS-NAT decoder for contextual modelling of token-level acoustic embeddings. The settings that achieve the best performance in this section (5 SADs and 2 MADs with NCM in both attention layers) are selected as default for subsequent experiments.

B. Error-based Sampling Alignment (ESA) Decoding

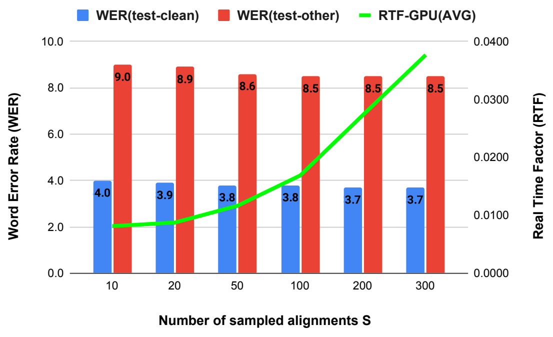
Viterbi-alignment is used in training, but not available during inference. To reduce the alignment mismatch between training and inference, we propose error-based sampling alignment (ESA). There are three important factors that can affect the WER performance of ESA: sampling threshold P, the number of sampled alignments S and scoring model for



(a) Effect of P using test-clean



(b) Effect of P using test-other



(c) Effect of the number of sampled alignments S

Fig. 3: WER performance of different values of the threshold P and the number of sampled alignments S in error-based sampling alignment (ESA) decoding. Real time factor (RTF) is evaluated using a V100 GPU with a batch size of one.

ranking alignments. We use the best decoder structure in the previous section (5SAD + 2MAD) to evaluate the performance of ESA decoding with different configurations.

Figures 3a and 3b show the results when the threshold P varies from 0.10 to 0.95 using the LibriSpeech test-clean and test-other data, respectively. The number of sampled alignments here is 10 and the scoring model is the AT baseline. We also include the WERs of the AT baseline and CASS-NAT decoding with best path alignment (BPA) as comparisons. As shown in the figures, ESA reaches the best performance when $P = 0.9$. A higher threshold indicates fewer sampled alignments, and thus no further improvement is observed. The threshold P is set to 0.9 in subsequent experiments.

Fig.3c shows the effect of the number of sampled alignments S in terms of WER and RTF on the test-clean and test-other data. As observed in the figure, by increasing the number of sampled alignments, the WER of ESA decoding improves but the improvement is small when S is greater than 50. Meanwhile, the RTF increases rapidly as S increases. Overall, $S = 50$ might be the most appropriate value to use as default in subsequent experiments.

Finally, various scoring models are compared. We consider using the AT baseline, neural language model (NLM) and n-gram language model for ranking the sampled alignments. NLM is a transformer-based model trained with the provided text in the LibriSpeech corpus. The n-gram models are also the ones provided in the LibriSpeech corpus. The results of ESA decoding together with that of BPA and ESA decoding are shown in Table II. As analyzed in Sec.III-C, BPA has an impressive speedup but the WER is much worse than the

TABLE II: Comparisons, in terms of WER and GPU speedup, of BPA, BSA, and ESA decoding. NLM is the transformer-based neural language model. 3-gram and 4-gram models are the ones offered in LibriSpeech.

	Scoring Model	dev-clean	dev-other	test-clean	test-other	GPU Speedup
AT baseline	-	3.4	8.1	3.6	8.0	1.00x
Oracle	-	2.1	5.5	2.2	5.3	39.6x
BPA	-	4.7	10.4	4.8	10.3	90.0x
BSA	-	3.8	8.8	3.9	8.8	0.90x
ESA	AT baseline	3.6	8.8	3.8	8.6	28.4x
	NLM	3.6	8.9	3.9	8.7	31.6x
	3-gram	5.4	11.4	5.8	11.4	32.6x
	4-gram	5.4	11.3	5.7	11.3	31.5x

AT baseline, and BSA can obtain a better WER but it is even slower than the AT baseline. By using neural network based scoring models, ESA can retain both the advantages of BPA and BSA. For example, using NLM as a scoring model, ESA can achieve a WER of 8.7% on the LibriSpeech test-other data and 31.6x speedup. The degradation in speedup compared to BPA originates from the ranking process. Using n-gram models for ranking alignments leads to worse WER compared to NLM in ESA because n-gram models are worse than NLM for language modelling. Another possible reason might be that the probability distribution of n-gram models is different from that of CASS-NAT decoder outputs, while for NLM it is similar. The reason why n-gram models have similar GPU speedup compared to NLM is that the scores of the sampled alignments cannot be obtained simultaneously. The best performing scoring model is the AT baseline, and thus the AT baseline is used in ESA decoding as default in the following experiments.

Note that because of the sampling process in ESA decoding, WER may be slightly different for different seeds of the random number generator. Fortunately, the randomness is small with a variance of around 0.5% relatively in our experiments.

C. Why does ESA Work? - An Analysis of the CTC Alignments

In this section, we analyze the CTC alignments obtained from different decoding strategies to understand why ESA can improve the performance of CASS-NAT. Two metrics are evaluated on the LibriSpeech test sets: mismatch rate (MR) and length prediction error rate (LPER). MR and LPER are measured between the alignments used in decoding and the oracle alignment, in which the blank and repetitive tokens are removed. The MR is the ratio of deletion and insertion errors compared to the oracle alignment, and substitution errors are not included because they do not change either the acoustic boundary for each token or the number of predicted tokens. If the number of tokens in an alignment is different from that in the oracle alignment, this alignment is considered as a case of length prediction error. LPER is the percentage of the utterances with length prediction errors.

Results of MR and LPER are presented in Table III. Note that the lower bound of WER (using oracle alignment) is 2.2% on the test-clean data assuming the transcriptions are available during decoding; this indicates that the framework is promising. When comparing the two metrics for different decoding strategies, we observe that the WER is more correlated with LPER than MR. This suggests that a correct estimation of the output length (the length of decoder input) is very important for NAT, which is also mentioned in [13]. The reason may be because CASS-NAT decoders have a stronger ability of correcting mistakes in CTC alignment by contextual token-level acoustic modelling when the length prediction is accurate. To further validate this assumption, we plot the length prediction error distribution and their corresponding WERs in Fig. 4. As seen from the figure, CASS-NAT achieves WERs that are lower than 2% when the length of the decoder input is estimated correctly, while the WERs are higher than 10% for those utterances with an absolute difference in length of more than 3 compared to the oracle alignment.

TABLE III: Comparisons of different decoding methods for CASS-NAT decoding. **Oracle**: Viterbi-alignment with ground truth. **MR**: mismatch rate; **LPER**: length prediction error rate (using word-piece as the modeling unit). **S**: the number of alignments sampled in ESA.

Decoding Method	S	WER (%)		MR (%)		LPER (%)	
		test-clean	test-other	test-clean	test-other	test-clean	test-other
Oracle	1	2.2	5.3	-	-	-	-
BSA	1	4.0	9.2	2.5	6.0	28.09	48.83
BPA	1	4.8	10.3	2.4	5.2	34.92	51.68
ESA	10	4.0	9.0	3.6	6.0	26.41	44.91
	50	3.8	8.6	3.8	6.3	25.46	43.08
	100	3.8	8.5	3.8	6.2	25.61	42.70
	300	3.7	8.5	3.8	6.3	25.53	42.67

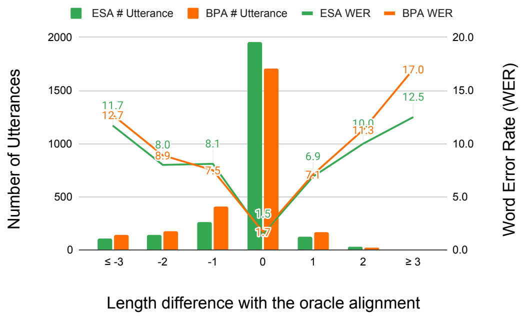


Fig. 4: The length prediction error distribution and their corresponding WERs using ESA(s=50) decoding on the test-clean dataset.

D. Improving the Training of CASS-NAT

Despite the use of ESA decoding method, the WER performance of CASS-NAT (8.6%) is still worse than that of the AT baseline (8.0%). In this section, we attempt to improve the WER performance of CASS-NAT by using various training strategies that include convolution-augmented encoder (ConvEnc.), intermediate CTC loss (InterCTC), convolution augmented decoder (ConvDec.), intermediate CE loss (InterCE) and trigger mask expansion (Mask Exp.). The results of various combinations of the training strategies are shown in Table IV.

First, we apply ConvEnc. and InterCTC to the autoregressive transformer to create a new AT baseline for fair comparisons. Note that, we have applied ConvDec. and InterCE to

TABLE IV: WERs of using the proposed training strategies on the LibriSpeech data. SpecAug is used in all configurations. WERR is the relative WER improvement compared to their corresponding baselines on the test-other data. KD represents using knowledge distillation from AT decoder outputs. Other strategies are the ones introduced in Section III-B. Bold faced numbers indicate the best WER results.

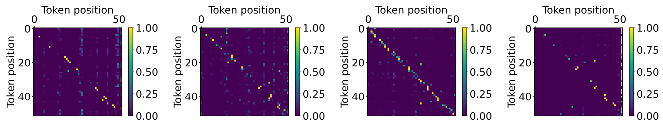
Model w/o LM	ConvEnc.	InterCTC	ConvDec.	InterCE	Mask Exp.	KD	dev-clean	dev-other	test-clean	test-other	*WERR*
AT							3.4	8.1	3.6	8.0	-
	✓						2.7	6.9	2.9	6.9	13.8%
	✓	✓					2.7	6.8	2.8	6.7	16.3%
CASS-NAT							3.6	8.8	3.8	8.6	-
	✓						3.0	7.3	3.2	7.5	12.8%
	✓	✓					2.9	7.4	3.1	7.3	15.1%
			✓				3.4	8.3	3.6	8.4	2.3%
			✓	✓			3.3	8.2	3.6	8.3	3.5%
	✓	✓	✓	✓			2.8	7.1	2.9	7.1	17.4%
					✓		3.6	8.9	3.8	8.7	-1.2%
			✓	✓	✓		3.3	8.3	3.6	8.1	5.8%
	✓	✓	✓	✓	✓		**2.7**	**7.1**	**2.9**	**7.0**	18.6%
						✓	3.6	8.8	3.8	8.5	1.2%

the AT baseline as well, but no improvements are observed, and thus their results are not included. When ConvEnc. and InterCTC are applied to CASS-NAT, similar improvements compared to their use for AT are observed. The improvements stem from their ability of capturing local details in the acoustic representations, and thus a stronger encoder is learned and it offers an accurate CTC alignment for implicit token-level acoustic embedding modelling in the decoder.

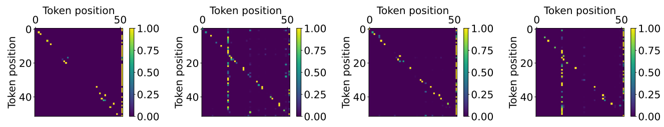
Second, when using ConvDec. and InterCE, the WER of CASS-NAT drops further to 7.1% on the test-other data, which is a 17.4% relative WER improvement over the CASS-NAT baseline. The use of ConvDec. and InterCE, however, is not as effective as ConvEnc and InterCTC, indicating that a stronger modelling of frame-level acoustic representations is more important than that of token-level acoustic representations (TAEs) in CASS-NAT. This is reasonable because TAEs are extracted from frame-level acoustic representations from the encoder. Interestingly, we observe a -1.2% relative WER reduction when the trigger mask expansion method is applied by itself. However, the method leads to WER improvements when it is used together with ConvDec. related strategies. The reason could be that expanding the acoustic boundary for each token might cause confusion for the decoder when the contextual modelling is not strong enough.

In addition, as shown in the last line of Table IV, applying knowledge distillation, which takes the AT output as a target for NAT training, does not result in WER improvements. Although knowledge distillation has been proven useful in non-autoregressive neural machine translation in [53], it does not seem to work for speech recognition possibly because there are no different outputs corresponding to the same input, while multiple translations of the source language exist in the target language for machine translation.

Finally, using all the training strategies, we achieve an 18% relative WER improvement compared to CASS-NAT baseline; this is only a 3% WER degradation compared to the best version of the AT baseline. In summary, we obtain a non-autoregressive speech transformer that performs close to its autoregressive counterpart with a significant GPU speedup.



(a) The $5^{th} \sim 8^{th}$ head of self-attention in the last SAD



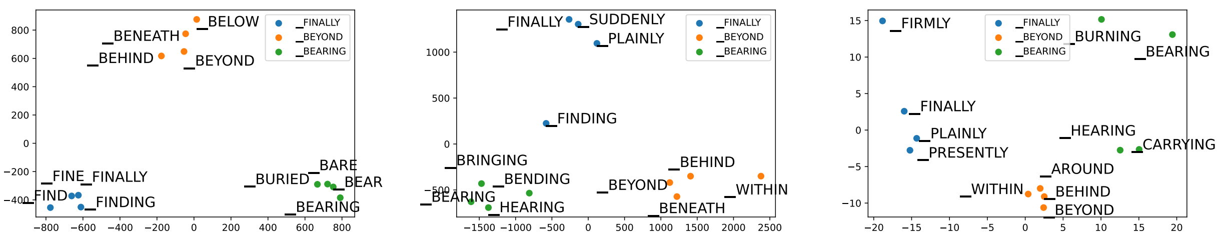
(b) The $5^{th} \sim 8^{th}$ head of self-attention in the last MAD

Fig. 5: Attention weight distributions of the $5^{th} - 8^{th}$ head in the last self-attention decoder (SAD) and mixed-attention decoder (MAD). The first utterance in the LiriSpeech train-clean-100 subset is used. The y-axis represents output tokens

E. Why does CASS-NAT Work? - An Analysis of the Decoder

In this section, we explain why CASS-NAT can achieve a good performance that is close to its autoregressive counterpart by analysing the following elements in CASS-NAT decoder: (1) attention weight distribution; and (2) acoustic-level acoustic representations.

First, to analyse the behaviour of the attention mechanism in CASS-NAT decoder, we choose the first utterance in the LibriSpeech train-clean-100 subset and plot the attention weight distribution in the last SAD and MAD blocks. The weights of the last 4 heads (8 heads in total) are shown in Fig.5. For self-attention weight distributions in SAD, we notice from the figure that most of the heads learn a monotonic alignment between token-level acoustic representations, indicating that each token relies on adjacent tokens. This is similar to the idea of word embedding using a continuous bag of words (CBOW) and skip-gram [54]. The monotonic alignment also shows the usefulness of the relative positional encoding because distant tokens with close semantic similarity



(a) TAEE output (b) SAD output (c) MAD output

Fig. 6: Visualization of token-level acoustic embeddings of three word pieces from outputs of TAEE, SAD and MAD, respectively. The embeddings are plotted using principle component analysis (PCA).

have low attention weights. For the attention weights in MAD, there exists a different behaviour in several subspaces of the attention computation, where outputs may rely on the same input (vertical line in Fig.5b).

As discussed in Section II-B2, we assume that language semantics can be captured by CASS-NAT decoder using token-level acoustic embeddings. To validate this assumption, we calculate three different token-level acoustic embeddings from TAEE, SAD and MAD for each token (word piece in our case). Specifically, token-level acoustic representations are first extracted from the first 5000 utterances in the train-clean-100 Librspeech subset, where each representation has its label in the form of a word piece. The embedding of each word piece is the average of the corresponding representations. After that, we randomly choose three word pieces and find the four closest ones (measured with cosine similarity distance) for each word piece. Using the same idea of visualizing word embeddings, the 12 selected embeddings are reduced to a 2-dimensional space using principal component analyses (PCA) and are then plotted. Examples of embeddings at different levels of the CASS-NAT decoder are shown in Fig.6. The figure suggests that the token-level acoustic embeddings learn not only acoustic similarities (**_BEAR** vs. **_BARE**), but also token-level semantic similarities (**_BENEATH** vs. **_BELOW**). This occurs, even though embeddings at different levels of the decoder provide different information. For example, higher layer (MAD) embeddings focus more on grammatical similarities (e.g. **_FIRMLY** and **_FINALLY**), which is similar to word embedding. This finding suggests the possibility of learning a joint speech and text embeddings in a common space. Even though there is no explicit language modelling, the CASS-NAT decoder is able to learn meaningful embeddings, which may explain why it has a similar performance to its AT counterpart.

F. Results on other datasets

Finally, we apply the proposed CASS-NAT to the other three datasets: Aishell (Mandarin speech), TED2 (English spontaneous speech), and MyST (child speech), to show the generalization ability of CASS-NAT. In addition, we explore the effect of different initialization schemes for the CASS-NAT encoder and measure the speedup given by CASS-NAT compared to AT baselines. All results are summarized in Table V. As can be observed from the table, a randomly initialized encoder for CASS-NAT achieves the best performance on the Aishell1, TED2 and MyST datasets, while the LibriSpeech dataset performs better with a pretrained encoder. The reason could be that we force the training iterations of CASS-NAT to be the same as the AT baseline for time considerations, as well as for a fair comparison, but LibriSpeech, as a large database compared to the others, needs more training iterations to obtain accurate alignments for the decoder. Hence, we suggest using a pretrained encoder for training CASS-NAT on large datasets.

With an appropriate encoder initialization, we now compare the results of CASS-NAT to the AT baselines and SOTA results using NATs. First, when compared to the AT baseline with greedy search decoding, CASS-NAT achieves better (on Aishell1, TED2, and MyST) or similar (on LibriSpeech) WERs performance, and has a 2.89x RTF speedup. When compared to the AT baseline with beam search decoding, the WERs performance on Aisehll and TED2 are still better, while the WERs on LibriSpeech and MyST are close to the baseline but with a 24x RTF speedup. Second, when compared to other NAT methods in the literature, we observe from the table that the proposed CASS-NAT achieves new state-of-the-art results on LibriSpeech, TED2, and MyST when no pretrained acoustic (e.g. Wav2vec2.0 [37]) or language models (e.g. BERT [59]) are used. To compare with the results that used extra training data (e.g. SSL with un-annotated data), we use the self-supervised pretrained models as the encoder for CASS-NAT to see whether it can outperform the SOTA results in Table V. Due to time and computational constraints, we obtain only the results on Aishell1 and MyST datasets. The results show that CASS-NAT can achieve comparable results to the best NATs in the literature [57], [58] with SSL pretraining.

VI. Conclusions

This paper presents a comprehensive study of the CASS-NAT framework introduced in [23], which utilizes alignments obtained from the CTC output space to extract token-level acoustic embeddings (TAEs), and regards the TAEs as substitutions of the word embeddings in autoregressive transformers (AT) to achieve parallel computation. During training, Viterbi-alignment is used to estimate the posterior

TABLE V: WER Results of CASS-NAT using different encoders as a starting point, including random initialization (Rand. Enc.), initialization from an encoder trained with CTC (CTC enc.), and initialization from a AT encoder (AT Enc.). AT baselines with greedy and beam search decoding are included together with SOTA results using non-autoregressive methods in the literature. "-" indicates that WER results have not been reported in the literature. RTF is evaluated with batch size of one on GPU. Bold-faced numbers are the best results for CASS-NAT and SOTA without self-supervised pretraining.

Model	Conditions	Inference Speed		LibriSpeech				Aishell1		TED2		MyST	
		RTF	Speed	dev clean	dev other	test clean	test other	dev	test	dev	test	dev	test
AT	greedy search	0.0387	1.00x	2.7	7.0	2.9	7.0	5.1	5.9	9.8	7.8	18.1	17.0
	beam search	0.3348	0.12x	2.7	6.8	2.8	6.7	4.7	5.1	8.2	7.6	16.5	16.2
Previous NAT results	w/o SSL			2.8	7.3	3.1	7.2 [24]	4.5	4.9 [17]	8.7	8.0 [55]	28.0	27.8 [56]
	w/ SSL			1.7	3.6	1.8	3.6 [22]	4.1	4.5 [57]	-	-	16.8	16.5 [58]
CASS-NAT (this work)	Rand. Enc.	0.0134	2.89x	3.3	8.4	3.6	8.3	**4.7**	**5.0**	8.4	**7.5**	**16.8**	**16.4**
	CTC Enc.			2.8	7.2	3.1	7.3	5.2	5.6	**8.0**	7.7	17.1	16.7
	AT Enc.			**2.7**	**7.1**	**2.9**	**7.0**	5.1	5.4	8.2	7.8	17.1	16.7
	HuBERT Enc.			-	-	-	-	4.2	4.5	-	-	16.0	15.6

probability, and various training strategies are used to improve the WER performance of the CASS-NAT. During inference, we investigate in depth an error-based sampling alignment (ESA) method that we introduced recently in [23] to reduce the alignment mismatch between training and inference. Detailed experiments show that: (1) a mixed-attention decoder (MAD) is important for reducing the WER; (2) the reason why ESA decoding works well is because it has a lower length prediction error rate; (3) convolution augmented encoder and decoder, intermediate loss and mask expansion can improve the WER of CASS-NAT, while knowledge distillation can not; and (4) TAEs have similar functionality to word embeddings, such as representing grammatical structures, indicating the possibility of learning semantics without a language model. As a result, CASS-NAT achieves new state-of-the-art results for NAT models on several ASR tasks with and without SSL pretraining. The datasets include English and Mandarin speech, read and spontaneous speech, and child and adult speech, showing the generaliztion ability of the proposed method. The performances of CASS-NAT are comparable, in relative terms, to AT with beam search decoding, but maintain a $\sim$24x speed up. Future work includes investigating the incorporation of language models using external data.

REFERENCES

[1] J. Li, Y. Wu, Y. Gaur, C. Wang, R. Zhao, and S. Liu, "On the comparison of popular end-to-end models for large scale speech recognition," *Proc. Interspeech 2020*, pp. 1–5, 2020.

[2] J. Li *et al.*, "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, 2021.

[3] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning (ICML)*, 2006, pp. 369–376.

[4] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[5] A. Graves, "Sequence transduction with recurrent neural networks," *International Conference of Machine Learning (ICML)*, 2012.

[6] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7829–7833.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 6000–6010.

[8] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5884–5888.

[9] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," in *International Conference on Learning Representations (ICLR)*, 2018.

[10] J. Lee, E. Mansimov, and K. Cho, "Deterministic non-autoregressive neural sequence modeling by iterative refinement," in *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1173–1182.

[11] C. Saharia, W. Chan, S. Saxena, and M. Norouzi, "Non-autoregressive machine translation with latent alignments," in *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1098–1108.

[12] W. Qi, Y. Gong, J. Jiao, Y. Yan, W. Chen, D. Liu, K. Tang, H. Li, J. Chen, R. Zhang *et al.*, "Bang: Bridging autoregressive and non-autoregressive generation with large scale pretraining," in *International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 8630–8639.

[13] N. Chen, S. Watanabe, J. A. Villalba, P. Zelasko, and N. Dehak, "Non-autoregressive transformer for speech recognition," *IEEE Signal Processing Letters (SPL)*, 2020.

[14] Y. Higuchi, S. Watanabe, N. Chen, T. Ogawa, and T. Kobayashi, "Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict," *Proc. Interspeech 2020*, pp. 3655–3659, 2020.

[15] Y. Bai, J. Yi, J. Tao, Z. Tian, Z. Wen, and S. Zhang, "Fast end-to-end speech recognition via non-autoregressive models and cross-modal knowledge transferring from bert," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 29, pp. 1897–1911, 2021.

[16] Z. Tian, J. Yi, J. Tao, Y. Bai, S. Zhang, and Z. Wen, "Spike-triggered non-autoregressive transformer for end-to-end speech recognition," *Proc. Interspeech 2020*, pp. 5026–5030, 2020.

[17] F. Yu, H. Luo, P. Guo, Y. Liang, Z. Yao, L. Xie, Y. Gao, L. Hou, and S. Zhang, "Boundary and context aware training for cif-based non-autoregressive end-to-end ASR," in *Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2021, pp. 328–334.

[18] N. Chen, P. Żelasko, L. Moro-Velázquez, J. Villalba, and N. Dehak, "Align-Denoise: Single-Pass Non-Autoregressive Speech Recognition," in *Proc. Interspeech 2021*, 2021, pp. 3770–3774.

[19] E. A. Chi, J. Salazar, and K. Kirchhoff, "Align-refine: Non-autoregressive speech recognition via iterative realignment," *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, pp. 1920–1927, 2020.

[20] 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 (ICML)*. PMLR, 2020, pp. 1403–1413.

[21] 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.

[22] E. G. Ng, C.-C. Chiu, Y. Zhang, and W. Chan, "Pushing the limits of non-autoregressive speech recognition," *in Proc. Interspeech*, pp. 3725–3729, 2021.

[23] R. Fan, W. Chu, P. Chang, and J. Xiao, "Cass-nat: Ctc alignment-based single step non-autoregressive transformer for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5889–5893.

[24] R. Fan, W. Chu, P. Chang, J. Xiao, and A. Alwan, "An Improved Single Step Non-Autoregressive Transformer for Automatic Speech Recognition," in *Proc. Interspeech 2021*, 2021, pp. 3715–3719.

[25] W.-N. Hsu, Y.-H. H. Tsai, B. Bolte, R. Salakhutdinov, and A. Mohamed, "Hubert: How much can a bad teacher benefit asr pre-training?" in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6533–6537.

[26] 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.

[27] P. Zhou, R. Fan, W. Chen, and J. Jia, "Improving generalization of transformer for speech recognition with parallel schedule sampling and relative positional embedding," *arXiv preprint arXiv:1911.00203*, 2019.

[28] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*, 2019, pp. 3286–3295.

[29] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, "Qanet: Combining local convolution with global self-attention for reading comprehension," in *International Conference on Learning Representations (ICLR)*, 2018.

[30] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *Proc. Interspeech 2020*, pp. 5036–5040, 2020.

[31] W. Han, Z. Zhang, Y. Zhang, J. Yu, C.-C. Chiu, J. Qin, A. Gulati, R. Pang, and Y. Wu, "Contextnet: Improving convolutional neural networks for automatic speech recognition with global context," *Proc. Interspeech 2020*, pp. 3610–3614, 2020.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 1–9.

[33] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6899–6903.

[34] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6224–6228.

[35] J. Lee, J. Kang, and S. Watanabe, "Layer Pruning on Demand with Intermediate CTC," in *Proc. Interspeech 2021*, 2021, pp. 3745–3749.

[36] Y. Wang, A. Mohamed, D. Le, C. Liu, A. Xiao, J. Mahadeokar, H. Huang, A. Tjandra, X. Zhang, F. Zhang *et al.*, "Transformer-based acoustic modeling for hybrid speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6874–6878.

[37] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *NeurIPS 2020*, 2020.

[38] 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 (JSTSP)*, vol. 11, no. 8, pp. 1240–1253, 2017.

[39] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6074–6078.

[40] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, "A new training pipeline for an improved neural transducer," *Proc. Interspeech 2020*, pp. 2812–2816, 2020.

[41] B. Yang, L. Wang, D. F. Wong, L. S. Chao, and Z. Tu, "Convolutional self-attention networks," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jun. 2019, pp. 4040–4045.

[42] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 2978–2988.

[43] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[44] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," in *20th Conference of the Oriental Chapter of the International Coordinating Committee on Speech Databases and Speech I/O Systems and Assessment (O-COCOSDA)*. IEEE, 2017, pp. 1–5.

[45] A. Rousseau, P. Deléglise, Y. Esteve *et al.*, "Enhancing the ted-lium corpus with selected data for language modeling and more ted talks." in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, 2014, pp. 3935–3939.

[46] W. Ward, R. Cole, D. Bolanos, C. Buchenroth-Martin, E. Svirsky, S. V. Vuuren, T. Weston, J. Zheng, and L. Becker, "My science tutor: A conversational multimedia virtual tutor for elementary school science," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 7, no. 4, pp. 1–29, 2011.

[47] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 66, 2018.

[48] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Sixteenth annual conference of the international speech communication association*, 2015, pp. 3586–3589.

[49] 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," *Proc. Interspeech 2019*, pp. 2613–2617, 2019.

[50] R. Fan, P. Zhou, W. Chen, J. Jia, and G. Liu, "An online attention-based model for speech recognition," *Proc. Interspeech 2019*, pp. 4390–4394, 2019.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *International Conference on Neural Information Processing Systems (NIPS)*, 2019, pp. 8026–8037.

[52] B. Zhang, H. Lv, P. Guo, Q. Shao, C. Yang, L. Xie, X. Xu, H. Bu, X. Chen, C. Zeng *et al.*, "Wenetspeech: A 10000+ hours multi-domain mandarin corpus for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6182–6186.

[53] C. Zhou, J. Gu, and G. Neubig, "Understanding knowledge distillation in non-autoregressive machine translation," in *International Conference on Learning Representations (ICLR)*, 2020.

[54] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *International Conference on Learning Representations (ICLR)*, 2013.

[55] 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 for speech-to-text generation," *Automatic Speech Recognition and Understanding (ASRU)*, pp. 47–54, 2021.

[56] R. Fan, Y. Zhu, J. Wang, and A. Alwan, "Towards better domain adaptation for self-supervised models: A case study of child asr," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 6, pp. 1242–1252, 2022.

[57] K. Deng, Z. Yang, S. Watanabe, Y. Higuchi, G. Cheng, and P. Zhang, "Improving non-autoregressive end-to-end speech recognition with pre-trained acoustic and language models," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8522–8526, 2022.

[58] R. Fan and A. Alwan, "DRAFT: A novel framework to reduce domain shifting in self-supervised learning and its application to children's ASR," in *in Proc. Interspeech 2022*, 2022, pp. 4900–4904.

[59] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pp. 4171–4186, 2019.