人工知能学会第二種研究会資料
AIチャレンジ研究会
SIG-Challenge-061-10

Blockwiseストリーミング音声認識と発話区間検出の統合

Integration of Blockwise Streaming Automatic Speech Recognition with Voice Activity Detection

周藤 唯[1*]　Muhammad Shakeel[1]　中臺 一博[2]　史 嘉彤[3]　渡部 晋二[3]
Yui Sudo[1]　Muhammad Shakeel[1]　Kazuhiro Nakadai[1]　Jiatong Shi[3]　Shinji Watanabe[1]

[1] (株) ホンダ・リサーチ・インスティチュート・ジャパン
[1] Honda Research Institute Japan Co., Ltd.
[2] 東京工業大学　[3] カーネギーメロン大学
[2] Tokyo Institute of Technology　[3] Carnegie Mellon University

Abstract: 本稿では，ストリーミング音声を入力とする音声認識アプリケーションに対応するため，Blockwise ストリーミング音声認識と発話区間検出の統合を扱う．近年，エンドツーエンド音声認識は実用的なシステムとして有望視されているが，ストリーミング音声入力に対応するためには以下の課題がある．1) 多くのエンドツーエンド音声認識モデルは音声入力があらかじめ短い発話に区切られていることを前提としているため，前段に発話区間検出モジュールが必要であり，システム全体のパラメータ数が増加する．2) 発話区間検出モジュールによる音声切り出しが適切でない場合，音声認識の性能が劣化する．3) 非発話区間が誤って発話として検出されると，性能劣化に加えて余分なデコードを行うための計算コストが増加する．そこで，本研究では，システム全体のパラメータを削減するため，Blockwise ストリーミング音声認識に発話区間検出ブランチを統合したモデルを提案する．また，ブロックごとに推定される発話区間検出結果を適切にデコード時に利用するため，Re-blocking 処理を提案する．提案手法は，既存の統合手法に対して，パラメータ数の増加を 1%未満に抑えながら，発話区間検出エラー率を 70.1%減少させた．さらに，同等のリアルタイムファクタ (RTF) を維持しつつ，文字誤り率 (CER) を 14.5%改善することができた．

1　はじめに

エンドツーエンド音声認識が実用的なシステムとして盛んに研究されている [1, 2, 3, 4, 5]．エンドツーエンド音声認識システムでは，従来の音声認識システムにおける音響モデルや言語モデル，発音辞書などのコンポーネントが不要になるため，システム構成を単純化することができる．システム構成が単純であることは，運用や保守を簡易化することが可能なため，実用性においては重要な特徴である．一方，ユーザにとっての実用性向上のための重要な要素は遅延時間である．ストリーミング音声を入力するアプリケーションでは，発話の完了を待たずに逐次的に音声認識結果が出力されることが要求されるため，ストリーミング処理が可能なエンドツーエンド音声認識モデルの研究が進められてきた．

最も単純なストリーミング音声認識の手法は，単方向 LSTM (Long Short Term Memory) を用いることである [6]．単方向 LSTM は未来の情報を必要としないため，発話の完了を待たずに認識処理を開始することができる．また，Transformer や Conformer といったオフラインベースの手法に対し，因果的な制約を設けることでストリーミング処理を可能にした Causal Transformer/Conformer も提案されている [7]．もう一つの方向性は，ブロック単位の処理を行うことで，Transformer や Conformer における self-attention の窓長を制限することである．このアプローチは，隠れマルコフモデルに基づくシステムや [8, 9]，RNN (Recurrent Neural Network) トランスデューサ [10, 11]，attention [12]，CTC (Connectionist Temporal Classification)/attention [13, 14, 15, 16] などで広く実現されている．ただし，これらのストリーミング音声認識モデルでは，入力音声があらかじめ発話ごとに分割されていることが前提となっているため，通常，音声認識の前段に発話区間検出モジュールが必要である．

発話区間検出に関しては，エネルギベースの手法 [17]，隠れマルコフモデル [18]，混合ガウスモデル [19] などの統計モデルを用いた手法が提案されている．また，近

*連絡先：(株) ホンダ・リサーチ・インスティチュート・ジャパン
〒 351-0188 埼玉県和光市本町 8-1
E-mail: yui.sudo@jp.honda-ri.com

年では，多層パーセプトロン [20]，LSTM [21, 22]，畳み込みニューラルネットワーク [23]，Transformer [24] を用いた深層学習ベースの方法も提案されている．発話区間検出は比較的計算量が少ないが，モジュールを追加することでシステム全体の複雑さが増してしまう．そこで，発話区間検出をエンドツーエンド音声認識に統合する試みもなされている [25, 26]．CTC ベースの発話区間検出 [27, 28] では，CTC が出力する blank ラベルを非音声セグメントと見なして発話区間の検出を行う．しかし，雑音などの非音声セグメントを明示的に blank ラベルに対応付けて損失関数を定義していないため，雑音を含む非音声に対して性能が劣化することが考えられる．

そこで，本研究では，発話区間検出モジュールの追加によるシステム全体のパラメータ増加を防ぐため，CTC/attention ベースの Blockwise ストリーミング音声認識 [16] に発話区間検出ブランチを統合したモデルを提案する．提案手法では，パラメータ数を最小限に抑えるため，音声認識と発話区間検出はエンコーダを共有する．また，ブロックごとに推定される発話区間検出結果を適切にデコード時に利用するための Re-blocking 処理を提案する．なお，本稿は，[29] の提案手法をもとに，既存の統合手法との比較実験を追加した．

2 ストリーミング音声認識

本節では，Transformer を用いた CTC/attention ベースの音声認識 [5] について述べた後，その拡張であり，提案手法に使用する Blockwise ストリーミング音声認識 [15, 16] について説明する．

2.1 Transformer を用いた音声認識

Transformer を用いた CTC/attention ベースの音声認識は，エンコーダ/デコーダ構造を持つ attention ベースの音声認識モデル [3] に，CTC を追加したモデルである [5]．

エンコーダは，畳み込み層，線形射影層，位置符号化層，Transformer ブロックから構成される．畳み込み層は，式 (1) のように長さ T の音響特徴列 $\boldsymbol{X} = [x_1, ..., x_T]$ を $\boldsymbol{u} = [u_1, ..., u_L]$ にダウンサンプリングする $(L < T)$.

$$\boldsymbol{u} = \mathrm{ConvSubsamp}(\boldsymbol{X}), \tag{1}$$

ダウンサンプリングされた特徴ベクトル列 $\boldsymbol{u}$ は，式 (2) のように，Transformer ブロックによって長さ L の隠れ状態ベクトル $\boldsymbol{h} = [h_1, ..., h_L]$ に変換される．

$$\boldsymbol{h} = \mathrm{TrEncoder}(\boldsymbol{u}), \tag{2}$$

Transformer ブロックは残差接続を持ち，multi-head self-attention 層，全結合層，layer 正規化層からなる．

デコーダは，エンコーダの出力する隠れ状態ベクトル $\boldsymbol{h}$ と過去に推定されたテキスト列 $\boldsymbol{y}_{s-1} = (y_0, ..., y_{s-1})$ を用いて，式 (3) に示すように，s 番目のテキスト y_s を再帰的に推定する．

$$y_s = \mathrm{TrDecoder}(\boldsymbol{h}, \boldsymbol{y}_{s-1}), \tag{3}$$

過去のテキスト列である $\boldsymbol{y}_{s-1}$ は，まず埋め込み特徴に変換される．埋め込み特徴と隠れ状態ベクトル $\boldsymbol{h}$ はデコーダに入力され，線形射影層，ソフトマックス関数を用いて y_s の予測確率が推定される．デコーダは self-attention，source-target attention，position-wise 全結合層から構成される．

CTC/attention ベースのモデルでは，上記に加えて全結合層およびソフトマックス関数から構成される CTC を持つ．CTC は，式 (2) によって出力される隠れ状態ベクトルを入力し，blank ラベルを含むテキスト確率を各時刻ごとに推定する．

2.2 Blockwise Transformer エンコーダ

音声がストリーミングで入力されるアプリケーションでは，音声認識モデルは発話の完了を待たずに逐次的に認識処理を行う必要がある．Transformer でこのようなオンライン処理を実現するため，Blockwise Transformer エンコーダ [15] では，式 (2) のエンコーダへの入力を式 (4) に示すようにブロック単位で行う．

$$\boldsymbol{u}_b = (u_{(b-1)L_{\mathrm{hop}}+1}, ..., u_{(b-1)L_{\mathrm{hop}}+L_{\mathrm{block}}}), \tag{4}$$

$\boldsymbol{u}_b$ は b 番目のブロックの特徴ベクトル列，L_{block}，L_{hop} はブロックサイズとホップ長を表す．b 番目の隠れ状態ベクトル $\boldsymbol{h}_b$ は，予め決められたブロックサイズ L_{block}，ホップ長 L_{hop} に基づいて，式 (5) に示すように，長さ L_{block} の隠れ状態ベクトルに変換される．

$$\boldsymbol{h}_b = \mathrm{BlockTrEncoder}(\boldsymbol{u}_b). \tag{5}$$

2.3 Blockwise ビームサーチ

Blockwise ストリーミング音声認識では，デコーダはエンコーダのブロック処理と同時に認識処理を行う必要がある [16]．attention ベース音声認識のオンラインビームサーチは，隠れ状態ブロック $\boldsymbol{h}_{1:b} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_b]$ が与えられた時における，もっとも確率の大きいテキスト列 $\hat{\boldsymbol{y}}$ を探索する問題として，式 (6) のように表される．

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y} \in \mathcal{V}^*}{\mathrm{argmax}}(\log(p(\boldsymbol{y}|\boldsymbol{h}_{1:b}))), \tag{6}$$

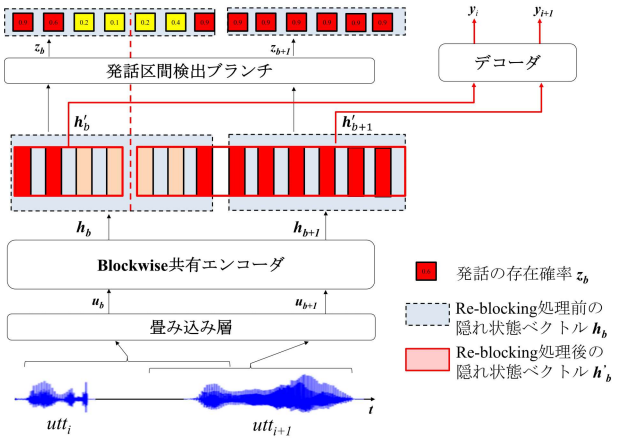

図 1: 提案手法の概要

式 (6) より，テキスト列の推定には，発話の開始から現在のブロックまでのすべての隠れ状態が必要となる．すなわち，発話が長くなるとデコードに要する時間は長くなる．したがって，隠れ状態ベクトル列はデコード前に適切に短い発話に分割し，隠れ状態ベクトルの履歴をリセットする必要がある．

3 提案手法

図 1 に提案手法の概要を示す．提案手法は，エンコーダ/デコーダ構造を持つ既存の CTC/attention ベースの Blockwise ストリーミング音声認識モデル [16] に加えて，発話区間検出ブランチを持つ．発話区間検出ブランチは，既存の音声認識モデルとエンコーダを共有するため，パラメータの増加を抑えることができる．

提案する統合モデルの学習は 2 段階で行う．まず，従来の音声認識モデルを単独で学習し，その後，音声認識モデルのパラメータを固定させて発話区間検出ブランチのみを学習する．推論時には，検出された発話区間情報に応じて適切に隠れ状態列を分割するため，Re-blocking 処理を用いてブロックサイズを調整する．

3.1 発話区間検出ブランチ

発話区間検出ブランチは全結合層とシグモイド関数で構成され，式 (5) の共有エンコーダの出力である隠れ状態ベクトル列 $\boldsymbol{h}_b$ をブロックごとに入力し，フレームごとの発話の存在確率 $\boldsymbol{z}_b = [z_1, ..., z_{L_{\text{block}}}]$ を推定する．時刻 t における発話の存在確率 z_t に対して，閾値 $p = 0.5$ 以上であれば，発話が存在するとみなす．また，発話中に存在する非常に短い非音声セグメントを誤検出しないように，連続する非音声セグメントに対する閾値 V を導入する．連続する非音声セグメントが閾値 V を超えた場合，そのセグメント列は非音声区間とみなされる．第 2.3 節で述べたように，デコード処理に用いる隠れ状態ベクトルが長くなると計算コストが増加してしまうため，非音声区間が検出された場合，現在ブロックまでに得られた隠れ状態ベクトルのデコード処理が完了した後，過去の隠れ状態ベクトルは破棄される．損失関数は，データセットから提供される発話区間情報をもとに目標出力 $\boldsymbol{z}$ を定義し，推定された出力 $\hat{\boldsymbol{z}}$ とのバイナリクロスエントロピーを用いる．

3.2 Re-blocking 処理

第 2.2 節で述べたように，Blockwise ストリーミング音声認識では隠れ状態ベクトル列はあらかじめ決められたサイズのブロック単位で処理される．しかし，非音声セグメントがブロックの途中に存在する場合，隠れ状態列を分割することができない．図 1 に，第 3.1 節で導入した連続する非音声セグメントの閾値が $V = 4$ の場合の具体例を示す．発話区間検出ブランチは，ブロック単位で L_{block} 個の隠れ状態ベクトル $\boldsymbol{h}_b$ を入力し，L_{block} 個の発話の存在確率 $\boldsymbol{z}_b$ を出力する．図 1 における $\boldsymbol{z}_b$ には，黄色で示す 4 個の連続した非音声セグメントが含まれている．連続する非音声セグメントが $V = 4$ 以上であるため，これは非発話区間とみなされるが，$\boldsymbol{z}_b$ の末尾のフレームには赤で示される次の発話冒頭の音声セグメントが含まれている．そのため，単純に長さ L_{block} のブロック単位で隠れ状態ベクトルを分割し，履歴をリセットしてしまうと次の発話の文脈が途切れてしまう．

そこで，デコード時には，検出された非音声セグメントの中心である C 番目の隠れ状態ベクトルを中心に，式 (7) に示すようにブロックサイズを変更 (Re-blocking 処理) する．

$$\boldsymbol{h}'_b = [h_{b_1}, ..., h_{b_C}]. \tag{7}$$

ここで，h_{b_1}，h_{b_C} は b 番目のブロックの先頭，C 番目の隠れ状態ベクトルを表す ($C < L_{block}$)．b 番目のブロックの中で，未使用の隠れ状態ベクトルは，次の式のように，次のブロックの隠れ状態ベクトル列 $\boldsymbol{h}_{b+1}$ に結合される．

$$\boldsymbol{h}'_{b+1} = \text{concat}([h_{b_{C+1}}, ..., h_{b_E}], \boldsymbol{h}_{b+1}), \tag{8}$$

h_{b_E} は，b 番目のブロックにおける末尾の隠れ状態ベクトルを表す．

4 評価実験

本節では，提案手法の発話区間検出の性能，音声認識性能向上に対する効果，計算時間の評価を行う．

4.1 実験条件

提案した統合モデルの入力は，サンプリング周波数16kHz，窓長512サンプル，ホップ長128サンプル，80次元のメルフィルタバンク特徴量を用い，SpecAugment [30] を適用した．エンコーダは，ストライドがそれぞれ2，3の2層の畳み込み層，512次元の線形射影層，位置符号化層，12層のTransformerブロックとlayer正規化層から構成される．デコーダは，2048個の隠れユニットを持つ6層のTransformerブロックで構成される．attentionの次元は256で，4つのmultihead self-attentionを持つ．第2.2節で述べたブロックサイズ L_{block} とホップ長 L_{hop} はそれぞれ40と16とした．発話区間検出ブランチには1層の全結合層を用いた．第3.1節で導入した連続する非音声セグメントの閾値は，V=10とした．音声認識モデル部のパラメータ数は30.3Mであったのに対し，統合した発話区間検出ブランチのパラメータ数はわずか0.8Kであった．

第1段階の音声認識モデル学習では，CTC，attentionの損失の重みをそれぞれ0.3，0.7として，マルチタスク学習を行った [4]．第1段階では，学習率0.005，ウォームアップステップ25,000で，音声認識モデル部のみをAdamを用いて40エポック学習した．第2段階では，学習率0.00001，ウォームアップステップ数10,000で，Adamを用いて発話区間検出ブランチを30エポック学習した．音声認識ツールキットとしてESPnetををを使用した [31]．評価には，日本語話し言葉コーパス (CSJ) [32] を使用した．CSJコーパスは，20分以上の長時間の録音を含んでいるため，ストリーミング音声認識の評価に適している．

4.2 発話区間検出タスクにおける評価

発話区間検出タスクにおいて，提案手法を外付けの発話区間検出モデルおよびCTCベースの発話区間検出 [27] と比較し，適切に発話区間を検出できることを確認した．

発話区間検出システムは，発話区間検出エラー率 (ER) を用いて評価した．ERは以下の式で求める．

$$ER = \frac{\sum_{t=1}^{T} F(t) + \sum_{t=1}^{T} M(t)}{\sum_{t=1}^{T} N(t)}, \tag{9}$$

$F(t)$ は誤検出された非音声セグメントの数，$M(t)$ は，検出されなかった音声セグメントの数を表す．

表1に発話区間検出に必要なパラメータ数とERを示す．外付けの発話区間検出モデルは4.45Mのパラメータを必要とするのに対し，提案手法は音声認識のエンコーダを共有するため，発話区間検出ブランチに必要なパラメータ数はわずか0.8Kであった．一方，CTCベースの発話区間検出は，音声認識モデルにおけるCTCによるblank出力を利用するため，パラメータの増加はない．しかし，雑音などの非音声セグメントを明示的にblankラベルに対応付けて損失関数を定義していないため，ERは大きかった．それに対し，提案手法では，追加パラメータ数をわずか1%未満に抑えつつ，CTCベースの発話区間検出手法よりもERを削減した．

表 1: 発話区間検出タスクにおける検出誤り率

手法	パラメータ数	eval1	eval2	eval3
外付け	4.45 M	4.9	3.9	5.4
CTC ベース	0	18.3	17.5	21.8
提案手法	0.8 K	5.5	4.6	6.9

4.3 音声認識タスクにおける評価

次に，提案手法を音声認識性能に対して，以下の手法との比較評価を行った．

- 手動発話切り出し: データセットが提供する，手動で切り出された発話時間情報に従って入力音声を分割した．これは，提案手法の上限値として扱う．
- 外付け発話区間検出: 外付けのBlockwiseストリーミングTransformerを用いた発話区間検出に基づいて，自動的に音声入力を分割した．
- 発話区間検出なし: 発話区間検出モジュールを用いず，隠れ状態ベクトル列の長さが L_{th}=300を超えた場合に自動的に音声入力を分割した．
- CTCベース発話区間検出 [27]: 連続するCTCのblank出力を非音声セグメントと見なし，自動的に分割した．この手法はもともとLSTMベースのエンコーダを使用していたが，公平な比較のためBlockwiseストリーミングTransformerに拡張した．
- 提案手法: 提案手法に基づき音声入力を自動分割した．また，3.2節で述べた発話区間検出の結果に基づいて，隠れ状態ブロックをRe-blocking化した．また，まれに長時間の発話があるため，すべての手法において，隠れ状態ベクトル列の長さが L_{th}=300を超えた場合にも自動的に音声を分割した．

認識性能の評価指標には，文字誤り率 (CER) を用いた．また，計算コストを検証するため，GPU (NVIDIA A100-SXM4-40GB) を用いて推論を行った際のリアルタイムファクタ (RTF) を測定した．

表 2: CER と RTF の比較

手法	eval1	eval2	eval3	RTF
手動発話切り出し	5.9	4.2	4.6	N/A
外付け発話区間検出	8.0	5.4	6.6	N/A
発話区間検出なし	10.1	7.4	7.9	0.55
CTC ベース	10.3	7.8	9.4	**0.40**
提案手法	**9.1**	**6.7**	**7.7**	**0.40**

4.3.1 実験結果

表 2 に CER と RTF を示す．まず，発話区間検出を用いない場合と提案手法を比較すると，すべての条件において CER と RTF を同時に削減することができた．発話区間検出なしの場合，発話の区切りとは無関係に一定間隔 (L_{th}=300) で音声が分割されるので，発話のコンテキストが分断されてしまうのに対し，提案手法では，発話区間検出ブランチを用いて適切に発話が分割されるため，CER が改善した．

RTF が改善した要因は，適切なタイミングで隠れ状態ベクトルの履歴をリセットすることで，計算量が削減されたことである．図 2 にデコード時の隠れ状態ベクトルのメモリサイズを示す．発話区間検出を用いない場合，非音声区間であってもデコード処理を続けてしまう．また，発話が非常に短かったとしても，一定時間 (L_{th}=300) の履歴を保持してしまう．それに対し，提案手法では，適切な発話の区切りで隠れ状態ベクトルの履歴をリセットすることができるため，RTF を削減することができた．

CTC ベースの手法と提案手法を比較すると，同等の RTF を維持したまま CER を改善することができた．表 1 に示すように，提案手法ではわずか 0.8K のパラメータしか追加していないため，計算コストへの影響はほとんど見られなかった．したがって，提案手法は，既存の CTC ベースの統合手法に対し，計算コストをほとんど増加させることなく，CER を改善することができたといえる．

しかし，提案手法は，外付けの発話区間検出を用いた場合よりも CER が悪化した．外付けの発話区間検出では，エンコーダも含めたモデル全体を同時に最適化したのに対し，提案手法は事前に学習された音声認識モデルのエンコーダを固定した状態で発話区間検出ブランチのみを最適化したため，性能が劣化したことが考えられる．今後の課題として，提案した統合モデルにおいても，2 段階学習ではなく発話区間検出ブランチと音声認識部を同時に最適化する手法の検討が必要であると考える．

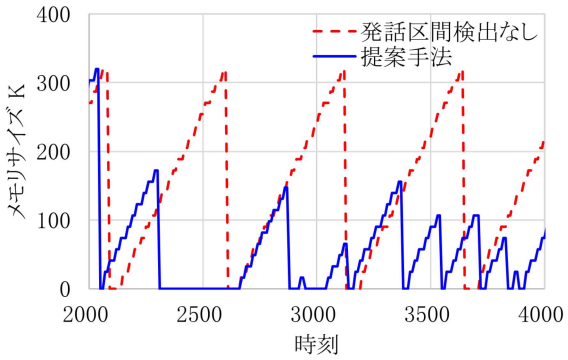

図 2: 隠れ状態ベクトルのメモリサイズ比較

表 3: Re-blocking の効果

手法	eval1	eval2	eval3
発話区間検出なし	10.1	7.4	7.9
Re-blocking なし	10.4	7.7	11.7
Re-blocking あり	**9.1**	**6.7**	**7.7**

4.3.2 Re-blocking の効果

表 3 に Re-blocking 処理の効果を示す．Re-blocking 処理を用いることで CER が改善したが，反対に Re-blocking 処理を用いない場合，むしろ発話区間検出なしの場合よりも CER は悪化した．これは，第 3.2 節で述べたように，Re-blocking 処理を用いずに事前に決められたブロック単位で音声を分割し、履歴をリセットしてしまうと，次の発話の文脈が途切れてしまうことがあるためである．したがって，Blockwise ストリーミング音声認識モデルにおいては，発話の文脈が不適切に分断されることを防止するため，提案した Re-blocking 処理が必要であることがわかった．

5 結論

本研究では，システム全体のパラメータを削減するため，Blockwise ストリーミング音声認識に発話区間検出ブランチを統合したモデル，および，ブロックごとに推定される発話区間検出結果を適切にデコード時に利用するための Re-blocking 処理を提案した．提案手法は，CTC ベースの既存の統合手法に対して，パラメータ数の増加を 1%未満に抑えながら，発話区間検出エラー率を 70.1%減少させた．さらに，同等のリアルタイムファクタ (RTF) を維持しつつ，文字誤り率 (CER) を 7.5%改善することができた．今後は，発話区間検出ブランチと音声認識部を同時に最適化することで，音声認識性能をさらに向上させる予定である．

参考文献

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

[2] A. Graves, "Sequence transduction with recurrent neural networks," in *Proc. ICML*, 2012.

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

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

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

[6] K. Rao, H. Sak, and R. Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer," in *Proc. ASRU*, 2017, pp. 193–199.

[7] B. Li, A. Gulati, J. Yu, T. N. Sainath, C.-C. Chiu, A. Narayanan, S.-Y. Chang, R. Pang, Y. He, J. Qin *et al.*, "A better and faster end-to-end model for streaming asr," in *Proc. ICASSP*, 2021.

[8] D. Povey, H. Hadian, P. Ghahremani, K. Li, and S. Khudanpur, "A time-restricted self-attention layer for ASR," in *Proc. ICASSP*, 2018, pp. 5874–5878.

[9] Y. Shi, Y. Wang, C. Wu, C.-F. Yeh, J. Chan, F. Zhang, D. Le, and M. Seltzer, "Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition," in *Proc. ICASSP*. IEEE, 2021.

[10] L. Lu, C. Liu, J. Li, and Y. Gong, "Exploring transformers for large-scale speech recognition," *Proc. Interspeech*, pp. 5041–5045, 2020.

[11] Y. Shi, V. Nagaraja, C. Wu, J. Mahadeokar, D. Le, R. Prabhavalkar, A. Xiao, C.-F. Yeh, J. Chan, C. Fuegen *et al.*, "Dynamic encoder transducer: A flexible solution for trading off accuracy for latency," *arXiv preprint arXiv:2104.02176*, 2021.

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

[13] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *Proc. ICASSP*, 2020, pp. 6074–6078.

[14] M. Li, C. Zorilă, and R. Doddipatla, "Head-synchronous decoding for transformer-based streaming asr," in *Proc. ICASSP*, 2021.

[15] E. Tsunoo, Y. Kashiwagi, T. Kumakura, and S. Watanabe, "Transformer ASR with contextual block processing," in *Proc. ASRU*, 2019, pp. 427–433.

[16] E. Tsunoo, Y. Kashiwagi, and S. Watanabe, "Streaming transformer ASR with blockwise synchronous beam search," in *Proc. SLT*, 2021, pp. 22–29.

[17] L. R. Rabiner and M. R. Sambur, "An algorithm for determining the endpoints of isolated utterances," *Bell System Technical Journal*, vol. 54, no. 2, pp. 297–315, 1975.

[18] D. Haws, D. Dimitriadis, G. Saon, S. Thomas, and M. Picheny, "On the importance of event detection for ASR," in *Proc. ICASSP*, 2016, pp. 5705–5709.

[19] A. Lee, K. Nakamura, R. Nisimura, H. Saruwatari, and K. Shikano, "Noice robust real world spoken dialogue system using GMM based rejection of unintended inputs," in *Proc. ICSLP*, 2004.

[20] N. Ryant, M. Liberman, and J. Yuan, "Speech activity detection on YouTube using deep neural networks," in *Proc. Interspeech*, 2013, pp. 728–731.

[21] T. Hughes and K. Mierle, "Recurrent neural networks for voice activity detection," in *Proc. ICASSP*, 2013, pp. 7378–7382.

[22] G. Gelly and J.-L. Gauvain, "Optimization of RNN-based speech activity detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 3, pp. 646–656, 2017.

[23] S. Thomas, S. Ganapathy, G. Saon, and H. Soltau, "Analyzing convolutional neural networks for speech activity detection in mismatched acoustic conditions," in *Proc. ICASSP*, 2014, pp. 2519–2523.

[24] W. Wang, X. Qin, and M. Li, "Cross-channel attention-based target speaker voice activity detection: Experimental results for M2MeT challenge," in *Proc. Interspeech*, 2022.

[25] S.-Y. Chang, B. Li, and G. Simko, "A unified end-pointer using multitask and multidomain training," in *Proc. ASRU*, 2019, pp. 100–106.

[26] B. Li, S.-y. Chang, T. N. Sainath, R. Pang, Y. He, T. Strohman, and Y. Wu, "Towards fast and accurate streaming end-to-end asr," in *Proc. ICASSP*. IEEE, 2020, pp. 6069–6073.

[27] T. Yoshimura, T. Hayashi, K. Takeda, and S. Watanabe, "End-to-end automatic speech recognition integrated with ctc-based voice activity detection," in *Proc. ICASSP*, 2020, pp. 6999–7003.

[28] Y. Fujita, T. Wang, S. Watanabe, and M. Omachi, "Toward streaming ASR with non-autoregressive insertion-based model," in *Proc. Interspeech*, 2021, pp. 3740–3744.

[29] Y. Sudo, S. Muhammad, K. Nakadai, J. Shi, and S. Watanabe, "Streaming Automatic Speech Recognition with Re-blocking Processing Based on Integrated Voice Activity Detection," in *Proc. Interspeech*, 2022, pp. 4641–4645.

[30] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[31] 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.

[32] K. Maekawa, "Corpus of spontaneous Japanese: Its design and evaluation," in *Proc. SSPR*, 2003.