



Streaming Intended Query Detection using E2E Modeling for Continued Conversation

Shuo-yiin Chang, Guru Prakash, Zelin Wu, Qiao Liang, Tara N. Sainath, Bo Li, Adam Stambler, Shyam Upadhyay, Manaal Faruqui, Trevor Strohman

Google Inc., U.S.A

{shuoyiin, guruprakash, zelinwu, tsainath, boboli, wildstone, strohman}@google.com

Abstract

In voice-enabled applications, a predetermined hotword is usually used to activate a device in order to attend to the query. However, speaking queries followed by a hotword each time introduces a cognitive burden in continued conversations. To avoid repeating a hotword, we propose a streaming end-to-end (E2E) intended query detector that identifies the utterances directed towards the device and filters out other utterances not directed towards device. The proposed approach incorporates the intended query detector into the E2E model that already folds different components of the speech recognition pipeline into one neural network. The E2E modeling on speech decoding and intended query detection also allows us to declare a quick intended query detection based on early partial recognition result, which is important to decrease latency and make the system responsive. We demonstrate that the proposed E2E approach yields a 22% relative improvement on equal error rate (EER) for the detection accuracy and 600 ms latency improvement compared with an independent intended query detector. In our experiment, the proposed model detects whether the user is talking to the device with a 8.7% EER within 1.4 seconds of median latency after user starts speaking.

Index Terms: end-to-end models, continued conversation

1. Introduction

Streaming speech recognition systems have been successfully used in many voice interaction applications e.g. voice assistant and dialog systems. In a typical voice-enabled environment, a predetermined hotword e.g., "OK Google", "Alexa", "Hey Siri" is used to activate a device in order to attend to the primary query which follows the hotword [1, 2, 3]. While hotword activation is commonly used for single query scenario, having to speak the hotword each time disrupts the flow of continued conversation. For example, the user may speak "OK Google, what's the weather like today?" After the system answers, the user has to speak "OK Google, what about tomorrow?" to get the forecast. This repeated hotword phrase can cause a cognitive burden on the user.

To get rid of the hotword on follow-on queries, it is crucial to build a classifier that identifies the utterances directed towards the device, referred to as intended-queries, while suppressing non-device directed utterances, referred to as unintended-queries e.g. speech to other individuals, thinking aloud or random speech. Here, we term this classifier as the intended-query (IQ) detector.

Figure 1 demonstrates the scenario of continued conversation. The system stays open for any follow-on utterances after a user speaks the first query using the hotword. Subsequent

queries are then filtered by intended-query detector that makes a series of binary decisions: whether to barge-in the system and send recognition results downstream for intended-queries or to discard for unintended-queries [4]. It is desirable to identify the intended-queries as soon as possible to respond to the user quickly while avoid accepting unintended-queries. Accepting unintended-queries would generate incorrect responses to the user, resulting in a very jarring experience. On the other hand, rejecting intended-queries or slow detection makes the system feel unresponsive.

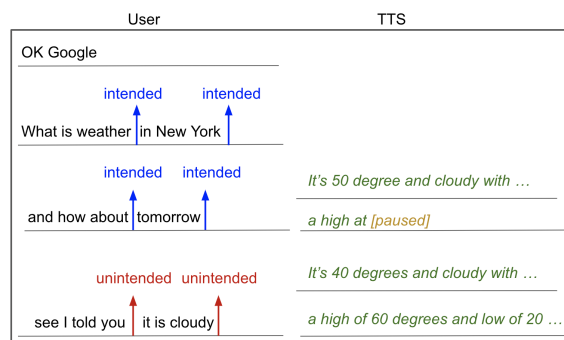


Figure 1: *Intended-query detection for follow-on queries*

Recent studies on intended-query detection have investigated acoustic-based approaches [5, 6] e.g. prosody, or using ASR decoder features [7, 8] e.g. lexical features or confidence. For example, in [4, 8, 9], word or subword embeddings from the top ASR hypothesis are explored as extra features in addition to acoustic and decoder features. The features from different sources are typically encoded by separate models and concatenated to a single feature vector per utterance. The utterance features are then fed through a fully connected network to compute the probability of the intended query. However, the utterance-level classifiers may need to consume a complete utterance to declare an IQ decision, which is undesirable for many streaming applications. For example, the system is expected to pause immediately if a user speaks to the device in the middle of system response as given by the example in Figure 1.

In contrast in this paper, we propose to build a sequence-to-sequence model that incorporates ASR and intended-query detection. This allows us to not only do ASR, jointly with intended-query detection, but also allows us to do the detection in a streaming fashion, which is important for latency sensitive applications.

E2E models look at folding different components of the speech recognition pipeline (i.e., the acoustic, pronunciation, and language models) into one neural network [10, 11, 12, 13, 14, 15]. Recent works have shown that folding further tasks into the model, for example endpointing, results in better WER and

latency tradeoff, compared to having separate tasks [16, 17]. In the spirit of joint optimization, in this paper, we proposed to build a joint ASR decoding and IQ detection with an E2E RNN-T model. To achieve this goal, we look to train the RNN-T model with additional tokens. Specifically, we incorporate the special symbols `<intended>` which indicates the intended queries and `<unintended>` for unintended queries as part of the expected label sequence e.g. "snooze alarm `<intended>` at 8:00". The RNN-T model then makes IQ decisions based on the IQ tokens. Thus, the model acts jointly as a decoder and an IQ detector in a streaming way.

We investigate the detection error tradeoff (DET) graph i.e. false accept and false reject rate to evaluate the quality of IQ detector. To evaluate the detection latency, we compute the latency, defined as the time between users starting speaking and the system declaring the IQ decision. The proposed E2E model achieves an 8.7% EER and a median latency of 1.4 seconds, which yields 22% and 600 ms improvement relative to the independent baseline detector.

2. Method

2.1. Baseline Systems

We investigate two external IQ detectors as baseline systems to compare with the proposed E2E IQ detector.

2.1.1. Acoustic IQ Detector

Figure 2a depicts the IQ detector based on acoustic features, referred as acoustic IQ detector. The acoustic IQ detector consists of long short term memory (LSTM) layers, which take typical acoustic features (i.e. log-mel filterbanks) as input and optimize for the frame-level targets of the two classes, intended or unintended. To obtain the frame-level targets, we simply label each frame of the entire utterances from Google’s assistant traffic as intended while label the frames of utterances from Google’s other speech applications e.g. YouTube as unintended. Thus, the acoustic IQ detector can extract distinctive acoustic patterns e.g. speech tempo to determine whether the audience for the utterance is likely talking to the device. To declare the final IQ decisions, we first obtain the frame-level decisions by thresholding the posterior of the intended class. Next, a state machine is applied on top of the frame-level decisions to smooth the distribution.

2.1.2. IQ Detector using Acoustic and Text embeddings

In addition to acoustic patterns, the spoken words also provide useful cues for differentiating between intended and unintended queries. Hence, a language model that directly models word sequences could improve the IQ detector. Figure 2b illustrates the IQ detector incorporating both the acoustic and word sequences following [4], termed as acoustic-text IQ detector.

To encode acoustic sequences, we take the hidden states of the last frame from the LSTM acoustic model to obtain a fixed length acoustic embedding per utterance. On the other hand, the language model encodes word sequences from top hypothesis of the speech recognizer. The hypothesis is first converted to a word embedding and then encoded with a convolutional neural network (CNN). The CNN uses max pooling to decrease the input and reduce the computational complexity of the network. Once the acoustic and text embeddings have been extracted, the embeddings are then concatenated and fed through a fully connected joint network to predict the probability of intended query

and unintended query.

The model incorporate both acoustic clues and text patterns to provide a more accurate detection accuracy compared with the IQ detector based on pure acoustic features. However, the acoustic and word sequences are encoded to utterance-level embeddings, thus the model could learn to predict the IQ decisions until it has seen the embeddings of a complete utterance when used in a streaming system. To address the problem, we propose to build a streaming E2E IQ detector built on ASR network in the following sections.

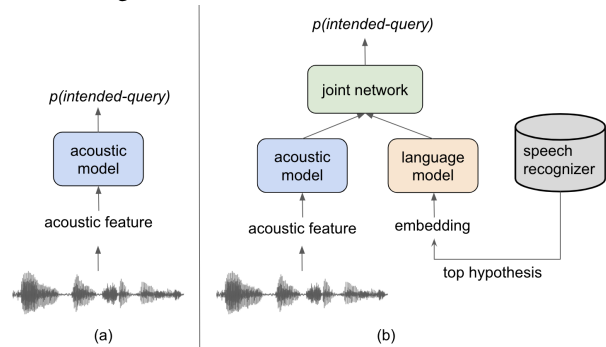


Figure 2: Baseline systems: (a) acoustic IQ detector and (b) acoustic-text IQ detector

2.2. E2E Model Labeling using Slot Filling

To build a sequence-to-sequence E2E ASR and IQ model, we insert `<intended>` to the label sequences of intended queries collected from Google assistant while `<unintended>` to unintended queries obtained from YouTube. A straightforward question with E2E ASR models that predict subword units (i.e., wordpieces), is how to insert special symbols i.e. IQ tokens, into the original label sequence. Our goal is to be able to detect `<intended>` as quickly as possible so the system and downstream NLU components can perform the appropriate action. In one extreme, if `<intended>` is inserted as the first symbol, while this allows a quick intent decision, the decision is unreliable as it is based only silence patterns before speaking and empty semantic history in streaming systems. In another extreme, if the symbol is appended at the utterance end, it could only emit the `<intended>` once finishing decoding the whole utterance, which is slow. Thus, our goal is to insert `<intended>` on incomplete transcriptions where a true intent has been initiated.

To achieve this goal, we exploit using slot filling and silence alignment as indicators of where to insert the IQ tokens. First, we extract the slots by parsing the transcriptions through an NLU model [18, 19] to obtain the span of the slots as shown in Fig. 3. Each slot represents a common semantic concept in the Google’s voice assistant traffic e.g. `<media_object>`, `<time_label>`, etc. Thus, the slot tags identify the parts of the transcriptions involving a complete semantic content. We then convert the closing tag of the slot into the `<intended>` tokens as Fig. 3. In addition, we also insert the `<intended>` or `<unintended>` at each silence segment to further force emitting IQ decisions quicker. These silence segments, indicating the user is pausing, are identified based on forced alignment.

2.3. RNN-T ASR and IQ detector

As illustrated in Fig. 4a, we use the RNN-T architecture while augmenting the label sequence with the IQ tokens as described

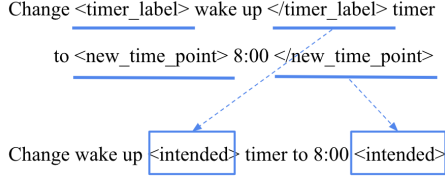


Figure 3: Labeling using slot filling

in Sec. 2.2 to build the E2E model. Thus, at each time step t , the model receives a new acoustic frame x_t and outputs a probability distribution over $y_t \in \{V \cup \langle \text{intended} \rangle \cup \langle \text{unintended} \rangle\}$, V being the wordpiece vocabulary and a blank symbol.

The RNN-T consists of 3 components: encoder, prediction network and joint network, which is analogous to acoustic model, language model and joint network in the baseline acoustic-text IQ detector (Fig. 2b) as described in Sec. 2.1.2. The encoder acts as the acoustic model. At each time step, the encoder receives typical acoustic features and outputs acoustic encodings. The recurrent prediction network obtains the embedding vectors from N previous (non-blank) labels [20] and produces the text encodings. Hence, the prediction network works as the language model of the acoustic-text IQ detector. The encoder and prediction network are then fed into the joint network that projects acoustic and text encodings to a fully-connected layer to computes the output logits.

However, simply introducing IQ tokens into the standard RNN-T model could degrade ASR quality. While the wordpiece history is useful for IQ prediction, the IQ tokens do not provide informative features for wordpieces prediction. Instead, including IQ tokens could decrease the context window as the prediction network attends to only a limited context [20]. In our experiments, we observed 4% relative WER increase.

To ensure that recognition quality is consistent with the standard E2E model, we adapt the model architecture by introducing separate joint networks, referred as ASR joint network and IQ joint network as illustrated in Fig. 4b. The ASR joint network provides the probability distribution over the wordpieces space as the conventional ASR while the IQ joint network is responsible for IQ decisions.

In training phrase, we perform two stages training strategy. First, we optimize the encoder, prediction network and the ASR joint network using the regular wordpieces label sequences for recognition quality. Next, we freeze the encoder and prediction network while initialize the IQ joint network with the ASR joint network. The IQ joint network is then fine-tuned with the expanded label sequence including wordpieces and IQ tokens to adapt the parameters with respect to additional loss due to IQ tokens insertion. Thus, the IQ joint network is able to predict distributions of IQ tokens given the existing acoustic and wordpieces label encodings. During inference, we rely on only the ASR joint network for beam search decoding over the word-piece space:

$$y^* = \arg \max_y \log P_{asr}(y | \mathbf{x}_{t-k}, \dots, \mathbf{x}_t, y_{u-N}, \dots, y_u) \quad (1)$$

where $\mathbf{x}_t, \dots, \mathbf{x}_{t-k}$ represents acoustic observations received by the Conformer encoder [21] with a context window of k and y_{u-N}, \dots, y_u stands for the wordpiece sequences. At each time step, the IQ joint network computes the the probability of the IQ tokens given the decoding paths obtained by ASR joint network

using Eq. 1. The posterior of IQ tokens can be expressed as:

$$P_{iq}(\langle \text{intended} \rangle | \mathbf{x}_{t-k}, \dots, \mathbf{x}_t, y_{u-N}, \dots, y_u), \quad (2)$$

$$P_{iq}(\langle \text{unintended} \rangle | \mathbf{x}_{t-k}, \dots, \mathbf{x}_t, y_{u-N}, \dots, y_u)$$

Thus, the system could make a series of IQ decisions at each time step by thresholding the posterior of $\langle \text{intended} \rangle$ obtained by Eq. 2. In order to emit the tokens early, we use the FastEmit [22] loss to encourage paths that outputs tokens earlier.

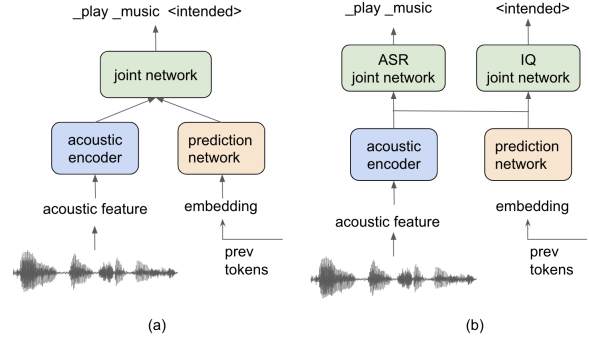


Figure 4: E2E model architectures using (a) conventional RNN-T and (b) RNN-T with additional joint network

3. Experiments

3.1. Datasets

The training data covers utterances of intended voice queries and unintended (nondevice-directed) voice transcriptions. The intended voice query data consists of around 15 million utterances collected from Google’s voice search and actions on various mobile platforms and Google Home. The unintended voice transcription includes around 10 million of utterances obtained from YouTube. The utterances are anonymized and hand-transcribed. In addition to the diverse training sets, multi-condition training (MTR) [23] are also used to further increase data diversity.

The evaluation set includes 53k utterances of proxy intended queries and 18k utterances of proxy unintended queries. The proxy voice queries are collected from 300 speakers. The proxy unintended queries covers 25 specific domains to include the scenarios of talking to other individuals, thinking aloud or random speech. Table 1 demonstrates the prompts for some proxy unintended query domains. The proxy intended queries include 18 domains with common voice actions, questions, search queries and queries in dialog scenarios.

Table 1: prompts of proxy unintended query.

Prompts of proxy unintended queries
Speak to your spouse about food/TV series or politics
Speak to your spouse when Assistant helps you well
Express you love the song (to yourself) Assistant played
Speak something that pops to you but on a very low voice

3.2. Model Description

The acoustic features used for the models are 128D log-mel filterbanks. Following [21], the RNN-T encoder network consists of 12 Conformer layers where each layer is of 512 dimension with a time-reduction layer after the 2nd layer. The Conformer layers consist of causal convolution and left-context attention

layers where 8-head attention is used in the self-attention layer and the convolution kernel size used is 15. The embedding prediction network [20] uses an embedding dimension of 320, and has 1.96M parameters. The ASR joint network and IQ joint network consists of a single feed-forward layer with 640 units. The RNN-T are trained with HAT factorization [24] to predict 4,096 word pieces [25] and two IQ tokens. The RNN-T model has 140M parameters. It is optimized by minimizing the RNN-T loss. The FastEmit [22] regularization with a weight of $5e-3$ is explored to improve the model’s prediction latency. We don’t use 2nd-pass model for the experiments here.

The baseline acoustic IQ detector used 3 LSTM layers with 64 units. The acoustic-text IQ detector follows the architecture in [4]. The language model combines the CNN word embedding layer with a 2 layer LSTM run over the utterances while the acoustic model consists of 3 LSTM layers with 64 units. The 100 units from the word embedding CNN’s max pooling layer are concatenated with the 64 units of the hidden state from the 2nd layer of the LSTM. These 164 units are then fed into two fully connected layers.

4. Experiments

4.1. Evaluation metrics

The DET curves are frequently used to describe a binary classification task. Here, we report the DET curve (false rejection against false accept) for the intended-query detection. Lower curves are better. The posterior of the intended-query is thresholded to obtain the IQ decision. A false accept would introduce incorrect responses to the user and extra computations for downstream components. A false rejection means the system is unresponsive. We use the EER to evaluate the performance of each approach. We also calculate the 50th latency and 90th latency of IQ detection where the latency is measured between the timestamp of start-of-speech and the timestamp of the intended query decision made. The latency is only calculated for the test sets of intended queries as the latency of declaring intended query is useful for quicker response.

4.2. Results

Table 2 demonstrates the EER of the baseline acoustic, acoustic-text IQ detector and the proposed E2E IQ detector. As shown in Table 2, the EER of the E2E IQ detector provides 22% relative improvement compared to the acoustic-text detector and is 49% better compared to the acoustic detector. As shown in Figure 5, the E2E IQ detector provides a consistently better FR/FA tradeoff compared to the two baseline models, especially for the operations points of FR below 10%, both of which degrade FA rapidly for the region.

For the latency metrics in Table 2, we observe the acoustic IQ detector provides the lowest latency to declare the IQ decisions despite higher EER compared with the other approaches. The acoustic-text detector has the highest 50th and 90th percentile latency. This implies that early incomplete audios and hypotheses often fail to trigger the IQ decisions where the long-tail examples require around 2.8 seconds to identify that users are talking to the device. Comparing to the acoustic-text detector, the proposed streaming E2E IQ detector could speed up the 90th percentile latency by 600 ms. The latency evaluation show that E2E IQ detector is only 60 ms slower to acoustic IQ for the 90th percentile latency while provides a much better EER.

Figure 6 reports the FR rates of each model for 5 selected domains of intended queries to investigate the errors. We eval-

uate the models with the operation points at EERs. The E2E IQ is better than both baseline systems for common actions e.g. music related actions, making a call, creating events, etc. Similar trends could be observed for the queries about asking facts e.g. ”how many metres in a mile” or questions ”why is the sky blue”. The E2E IQ shows consistent gains over the other systems except for a specific domain (undo action) that has fewer training examples where the acoustic IQ is more robust.

Table 2: *EER and latency.*

model	EER (%)	50th latency	90th latency
Acoustic IQ	17.2	1200 ms	2160 ms
Acoustic-Text IQ	11.2	1860 ms	2820 ms
E2E IQ	8.7	1440 ms	2220 ms

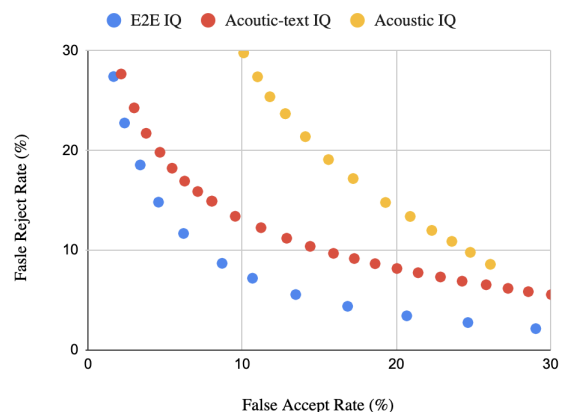


Figure 5: *DET for acoustic IQ detector, acoustic-text IQ detector and E2E IQ detector*

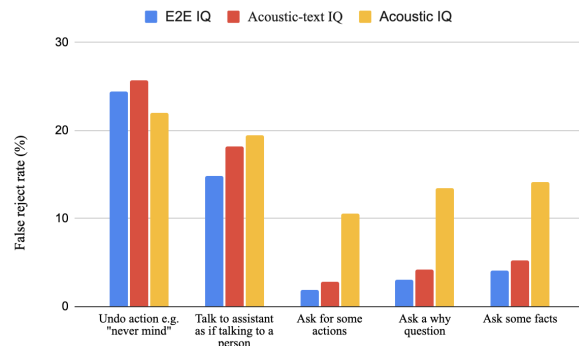


Figure 6: *FRs of specific domains*

5. Conclusions

In this work, we propose a sequence-to-sequence model that incorporates ASR and intended-query detection into a unified E2E RNN-Transducer. By joint modeling, the E2E model allow us to do intended-query detection and ASR jointly in a streaming fashion to achieve a fast and accurate intended-query detection. By comparing to a separate system, results and analyses show that the E2E model reduces the ERR from 11.2% to 8.7% and provides 600ms latency improvement.

6. References

- [1] Y. Huang, T. Z. Shabestar, and A. Gruenstein, “Hotword cleaner: dual-microphone adaptive noise cancellation with deferred filter coefficients for robust keyword spotting,” in *Proc. ICASSP*, 2019.
- [2] R. Tang and J. Lin, “Deep residual learning for small-footprint keyword spotting,” in *Proc. ICASSP*, 2018.
- [3] M. Wollmer, B. Schuller, and G. Rigoll, “Keyword spotting exploiting long short-term memory,” in *Speech Commun., vol. 55, pp. 252–265*, February 2013.
- [4] N. Howard, G. Simko, C. Parada, R. Kalyanasundaram, and S. Vasudevan G. Prakash, “Utterance classifier,” <https://patentimages.storage.googleapis.com/c0/d8/bf/540d2c6cd58a61/US20200349946A1.pdf>, U.S. Patent US 20200349946A1, November 2020.
- [5] A. Norouzian, B. Mazouze, D. Connolly, and D. Willett, “Exploring attention mechanism for acoustic-based classification of speech utterances into system-directed and non-system-directed,” in *ICASSP*, 2019.
- [6] E. Shriberg, A. Stolcke, and S. Ravuri, “Addressee detection for dialog systems using temporal and spectral dimensions of speaking style,” in *Interspeech*, 2013.
- [7] H. Lee, A. Stolcke, and E. Shriberg, “Using out-of-domain data for lexical addressee detection in human-human-computer dialog,” in *NAACL-HLT*, 2013.
- [8] S. Mallidi, R. Maas, K. Goehner, and B. Hoffmeister A. Rastrow, S. Matsoukas, “Device-directed utterance detection,” in *Interspeech*, 2018.
- [9] C. Huang, R. Maas, S. Mallidi, and B. Hoffmeister, “A study for improving device-directed speech detection toward frictionless human-machine interaction,” in *Interspeech*, 2019.
- [10] J. Li, Y. Wu, Y. Gaur, et al., “On the Comparison of Popular End-to-End Models for Large Scale Speech Recognition,” in *Proc. Interspeech*, 2020.
- [11] Y. He, T. N. Sainath, R. Prabhavalkar, et al., “Streaming End-to-end Speech Recognition For Mobile Devices,” in *Proc. ICASSP*, 2019.
- [12] C.-C. Chiu, T. N. Sainath, Y. Wu, et al., “State-of-the-art Speech Recognition With Sequence-to-Sequence Models,” in *Proc. ICASSP*, 2018.
- [13] S. Kim, T. Hori, and S. Watanabe, “Joint CTC-attention based end-to-end speech recognition using multi-task learning,” in *Proc. ICASSP*, 2017.
- [14] J. Li, R. Zhao, H. Hu, and Y. Gong, “Improving RNN transducer modeling for end-to-end speech recognition,” in *Proc. ASRU*, 2019.
- [15] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, “A new training pipeline for an improved neural transducer,” in *Proc. Interspeech*, 2020.
- [16] S. Chang, R. Prabhavalkar, Y. He, T.N. Sainath, and G. Simko, “Joint Endpointing and Decoding with End-to-End Models,” in *Proc. ICASSP*, May 2019.
- [17] B. Li, S.-Y. Chang, T. N. Sainath, et al., “Towards fast and accurate streaming end-to-end ASR,” in *Proc. ICASSP*, 2020.
- [18] P. Kaliamoorthi, “Advancing nlp with efficient projection-based model architectures,” <https://ai.googleblog.com/2020/09/advancing-nlp-with-efficient-projection.html>, September 2020.
- [19] C. Xiong, R. Socher, J. Bradbury, S. Merity, “Quasi-recurrent neural networks,” in *ICLR*, 2017.
- [20] R. Botros and T.N. Sainath, “Tied & reduced rnn-t decoder,” in *Proc. Interspeech*, 2021.
- [21] A. Gulati, J. Qin, C.-C. Chiu, et al., “Conformer: Convolution-augmented Transformer for Speech Recognition,” in *Proc. Interspeech*, 2020.
- [22] J. Yu, C.-C. Chiu, B. Li, et al., “FastEmit: Low-latency Streaming ASR with Sequence-level Emission Regularization,” in *Proc. ICASSP*, 2021.
- [23] C. Kim, A. Misra, K. Chin, T. Hughes, A. Narayanan, T. N. Sainath, and M. Bacchiani, “Generated of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in Google Home,” in *Proc. Interspeech*, 2017.
- [24] E. Variani, D. Rybach, C. Allauzen, and M. Riley, “Hybrid Autoregressive Transducer (HAT),” in *Proc. Interspeech*, 2020.
- [25] M. Schuster and K. Nakajima, “Japanese and Korean voice search,” in *Proc. ICASSP*, 2012.