CACHING NETWORKS: CAPITALIZING ON COMMON SPEECH FOR ASR

Anastasios Alexandridis, Grant P. Strimel, Ariya Rastrow
Pavel Kveton, Jon Webb, Maurizio Omologo, Siegfried Kunzmann, Athanasios Mouchtaris

Amazon.com
{aanast,gsstrime}@amazon.com

ABSTRACT

We introduce Caching Networks (CachingNets), a speech recognition network architecture capable of delivering faster, more accurate decoding by leveraging common speech patterns. By explicitly incorporating select sentences unique to each user into the network's design, we show how to train the model as an extension of the popular sequence transducer architecture through a multitask learning procedure. We further propose and experiment with different phrase caching policies, which are effective for virtual voice-assistant (VA) applications, to complement the architecture. Our results demonstrate that by pivoting between different inference strategies on the fly, CachingNets can deliver significant performance improvements. Specifically, on an industrial-scale, VA ASR task, we observe up to 7.4% relative word error rate (WER) and 11% sentence error rate (SER) improvements with accompanied latency gains.

Index Terms— automatic speech recognition, latency, streaming, end-to-end, personalization

1. INTRODUCTION

Automatic Speech Recognition (ASR) systems are becoming evermore incorporated into our daily lives and routines. In particular, with smart phones, speakers and tablets seemingly always within reach, there are more opportunities to interface with the technology. As a result, for widespread adoption, these systems are required to be reliable for the end user – driving the development of ASR modeling approaches that not only yield high accuracy decoding but can process audio signals in near-real time.

To that end, modern techniques have broadly focused on developing fully-neural, end-to-end architectures [1, 2, 3] to address considerations such as predictive performance [4], footprint [5], latency [6] and even model update cost [7]. These architectures are popular because in addition to overcoming such constraints, they also often better integrate with modern hardware enabled on edge-devices [6]. The recurrent neural network-transducer (RNN-T) [1], and its sequence-transducer variants [8, 9, 10], is a favored choice by virtue of its modeling capacity but largely due to its streaming functionality. The streaming capability, producing words as close to after they are spoken as possible, is essential for delivering low-latency responses from the VA, especially since ASR decoding is often just the frontier of a cascaded VA workflow, where the decoded results are forwarded to composite downstream systems, such as natural language understanding [11, 12], entity resolution [13] and text-to-speech models [14]. Thus, speedy and accurate decoding leads to an overall responsive system which reacts to users' requests with the desired intent in a dependable manner.

The first two authors contributed equally.

In this work we focus on both accuracy and latency improvements for VA ASR systems with an architecture enabled to dynamically focus on select sentences. Predictive performance for ASR models is most commonly measured in terms of WER and SER. Latency, on the other hand, is a more nuanced concept. As study [15] categorizes, *user perceived latency* (UPL), the time difference between the end of the user's speech and when ASR completes its decoding, materializes in three forms: algorithmic latency, compute latency and end pointing latency (EPL). Algorithmic latency, which measures the duration of each audio chunk and any additional lookahead frames required by the model, often underestimates true UPL. Compute latency emerges from the inference complexity of the neural ASR model, typically measured by floating point operations, in relation to the processing speed of the executing hardware. Many approaches seek to address this cost by reducing the inference operations required [16, 17, 18, 19], but compute latency is a key driver of UPL primarily when working within a significantly resource constrained setting. On the other hand, EPL, defined as the frame level difference between when the end-of-speech token (<\s>) is emitted in the finalized recognition result and when a user finishes speaking, is invariably a consideration regardless of platform. Furthermore, [15] suggests EPL, and token emission latency more broadly, will typically be the dominant drivers of observed UPL. We adopt this view here as well and use EPL as our approximation for UPL.

In this work, we design an open vocabulary speech recognition architecture to deliver additional latency and accuracy benefits by leveraging usage patterns of VA-type applications. The intuition of our design is that while VA traffic forms a long tail distribution, most utterances occur within the head – only a select set of utterances and domains are responsible for a disproportionate share of traffic. These phrases are common command-and-control type utterances, such as 'turn off the tv', but also include queries such as 'how much time is left on my timer'. Furthermore, in a user specific deconstruction, the global long tail distribution is composed of many differing distributions unique to each individual. For example, while the utterance 'shuffle my classical music playlist' may be considered a relatively rare utterance in the global distribution, a particular person might direct this to a VA each day, resulting in the designation as a head utterance for that user. Additionally, use cases like *Alexa Routines* [20], allowing users to set any series of actions to a customized phrasing, e.g. *when I say X do Y*, further showcases that users have unique common phrases to them but which are rare globally. With this motivation, we present CachingNets, where we train an end-to-end neural ASR architecture which can pivot its inference procedure midstream between traditional transducer-based beam search decoding and multiclass classification on a focused subset of phrases, i.e. a cache. This dual mode capability has the potential for improving both WER and latency for ASR models. The accuracy improve-

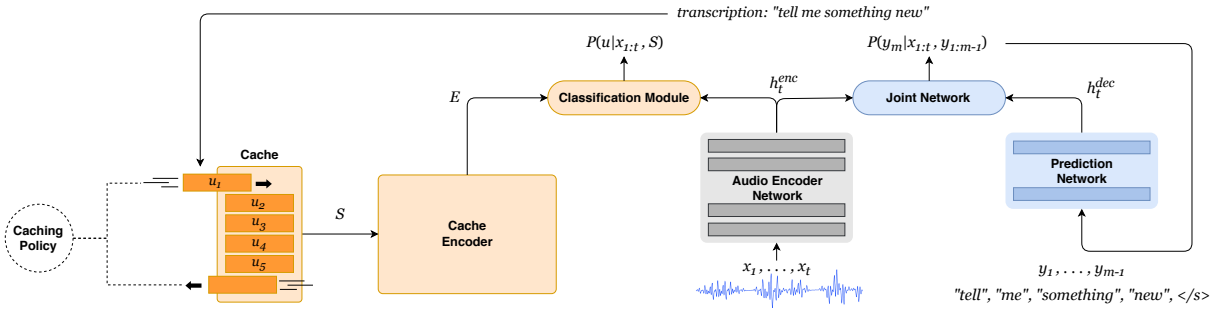

Fig. 1. The CachingNet architecture. Multitasking occurs on the representations produced by the audio encoder of a sequence-transducer model. An augmented classification module uses the acoustic representations to match the utterance against a cache S of predefined transcriptions. A caching policy is applied to strategically select which phrases populate the cache, and a cache encoder transforms S into a representation E ingested by the classification module.

ment is driven by the simpler decoding on the cached common transcriptions, while a latency improvement is achieved from reduced processing when an early match is found. Our work also draws on a strong, active line of research making end-to-end models robust to the long tail through n-gram level contextualization and biasing methods [6, 21, 22, 23, 24]; however, our aim is to operate over whole sentences by transforming them into head distributions for exploitation with alternate inference mechanics for improved speed and accuracy. Our contributions include the CachingNets architecture construction, applying central tenets of caching to strategically select a focus set of sentences, and experimental results that demonstrate the effectiveness of the approach.

2. A CACHING ASR ARCHITECTURE

At a high level, the CachingNet is implemented by multitasking an augmented speech recognition model. For our design we leverage the transducer-based architectures, namely RNN-T and its variants. This architecture consists of three main components: the audio encoder $\mathcal{F}$, prediction network $\mathcal{G}$, and joint network $\mathcal{J}$. $\mathcal{F}$ takes a vector of pre-processed audio feature frames $x_{1:t}$ as input and returns output h_t^{enc} for $t \leq T$. $\mathcal{G}$ takes a vector embedding of previous non-blank wordpiece labels $y_{1:m}$ of the transcription and returns intermediate output h_m^{dec} for $m \leq M$. Outputs from the encoder and decoder are passed to the joint network: $\mathcal{J}(h_t^{\text{enc}}, h_m^{\text{dec}}) = \boldsymbol{W}_{\text{vocab}}\psi(\boldsymbol{W}_{\text{enc}}h_t^{\text{enc}} + \boldsymbol{W}_{\text{dec}}h_m^{\text{dec}})$. The conditional output distribution $p_{t,m} = P(\hat{y}_{m+1}|x_{1:t}, y_{1:m})$ is obtained by applying a softmax to the result of $\mathcal{J}$.

The CachingNet design leverages the representations produced by $\mathcal{F}$ for multitasking, funneling its output to both the traditional joint network and a new component, the *classification module*. The classification module $\mathcal{C}$ accepts streaming hidden representations from $\mathcal{F}$ and treats decoding as a classification task at each frame over a set S of N predefined, *cached* utterance transcripts. These transcripts will be encoded and form the secondary, but fixed, input of $\mathcal{C}$.

2.1. Cache Encoder

Each classification output of the new augmented architecture is tied to one of N predefined transcripts. We refer to this set of cached utterances as the support set $S = (u_1 \ldots u_N)$. The cache encoder $\mathcal{E}$ serves the role of taking transcription strings constituting the support set S and encoding them into a representation used in the architecture's classification module.

We apply a transformer [25] based encoder $\mathcal{T}$ to operate over each transcription. We use a 6 transformer block encoder of 1024 dimension with 8 multi-headed attention (MHA) heads (each at size 128) and a 512 point-wise feed-forward hidden dimension. Input sentences are tokenized with the same word-piece tokenizer used for the RNN-T model and a standard embedding layer is learned with positional encoding applied before the transformer blocks. The outputs of the transformer encoder are average-pooled across tokens producing a single 1024 representation e_i for each entry u_i in S.

$$\begin{aligned}\mathcal{E}(S) &= (\text{Avrg}(\mathcal{T}(u_1)), \ldots, \text{Avrg}(\mathcal{T}(u_N))) \\ &= (e_1, \ldots, e_N) \\ &= E\end{aligned}$$

2.2. Classification Module

The classification module operates on the transcription representations computed by the cache encoder and the audio encoder output vector h_t^{enc}. We use an MHA variation that we denote as a multi-headed scoring mechanism (MHS), where similarity is computed between our query h_t^{enc} and keys E, but only scores are forwarded for further processing without application to a value list. The operation for a single head j is defined as:

$$\text{score}_j(E, h_t^{\text{enc}}) = (\boldsymbol{W}_q h_t^{\text{enc}})^\top (\boldsymbol{W}_k E),$$

using simple dot-product attention as our scoring function. The scores outputted by the N_H heads are concatenated and passed though a dense layer:

$$\begin{aligned}\text{MHS}(h_t^{\text{enc}}, E) &= \boldsymbol{W}_o [\text{score}_1 \cdots \text{score}_{N_H}] \\ &= \text{MHS}_t.\end{aligned}$$

The MHS result is concatenated with h_t^{enc} and fed through a final accumulating LSTM to aggregate scores and smooth the predictions over the sequence:

$$h_t^{\text{cls}} = \text{LSTM}\left([h_t^{\text{enc}}; \text{MHS}_t], h_{t-1}^{\text{cls}}\right).$$

Finally, the result is projected to dimension $N+1$ and then a softmax is taken:

$$\mathcal{C}(h_t^{\text{enc}}, E) = \text{Softmax}\left(\boldsymbol{W}_p h_t^{\text{cls}}\right) = c_t.$$

We interpret $c_t^{(i)}$ as the conditional probability that the utterance's transcription y matches cached transcription u_i at time frame t, i.e. $c_t^{(i)} = P(y = u_i|x_{1:t}, S)$ with $c_t^{(N+1)} = P(y \notin S|x_{1:t}, S)$.

2.3. Loss Function

Training the CachingNet architecture uses examples expressed as 4-tuples: $(x_{1:T}, y_{1:M}, S, z)$, where value z takes on the the class index

i if $u_i = y$ and $N + 1$ otherwise. We use two loss functions for the multitask learning: $\mathcal{L}_{\text{RNN-T}}$, which is the sequence-transducer loss using teacher forcing, and $\mathcal{L}_{\text{cls}}$ for the caching classification task. We combine these for the final loss function with scaling parameter λ weighting the relative contributions constituting the combined loss.

$$\mathcal{L} = \mathcal{L}_{\text{RNN-T}}\left(p_{T\times M}, y_{1:M}\right) + \lambda \mathcal{L}_{\text{cls}}\left(c_{1:T}, z\right)$$

We use $\lambda = 1$ and our implementation of $\mathcal{L}_{\text{cls}}$ is a length normalized cross entropy loss across all timesteps.

$$\mathcal{L}_{\text{cls}}\left(c_{1:T}, z\right) = -\frac{1}{T}\sum_{t=1}^{T} w_t \log c_t^{(z)}$$

where w_t is a weighting function of the loss contribution of timestep t.

While in principle, the classification outputs should be well calibrated upon convergence with $c_t^{(i)} \approx P(y = u_i | x_{1:t}, S)$ at each timestep t using a constant weighting of $w_t = 1$, an emphasis for high precision classification where it is most required near the end of the utterance is not accounted for. As a result, we consider a weighting scheme to emphasize accuracy near the tail where the inference decisions at runtime are more likely to occur. We propose the following:

$$w_t = (1 - \alpha)\frac{1}{1 + e^{\gamma(\beta T - t)}} + \alpha,$$

which is a logistic weighting with minimum weight α, inflection point at βT, and growth rate γ. Using this back-loaded weighting scheme, we find the model training to be significantly more stable, with faster, superior convergence as opposed to naive weighting. We applied $\alpha = 1/32$, $\beta = 2/3$, and $\gamma = 1$ with success but observe a broad range of parameters are viable.

2.4. Inference

The CachingNet decoding operates in a dual-mode, conducting inference from both the outputs of the transducer joint network and classification module simultaneously. The audio encoder operates in streaming mode (left-to-right) emitting per-frame acoustic embeddings. These representations are fed both to the classification module and joint network for standard RNN-T beam search. The classification module makes running decisions on-the-fly. If the classification result reaches a saturated enough score (i.e. $c_t^{(i)} \geq \theta$ for chosen parameter θ) before end-of-utterance (<\s>) is emitted from beam search decoding, the utterance transcript is recognized as u_i with high confidence and we short circuit RNN-T. Otherwise, RNN-T beam search completes decoding as usual and the classification result is discarded. [1] Note that just as RNN-T decoding terminates when <\s> emission reaches critical threshold, so does the classification module, except the classification termination is accompanied with a full, prepopulated transcription.

3. CACHING POLICIES

One of the most important design choices of any system utilizing a cache is that of the *caching policy*. The caching policy is the strategy used to populate and evict entries from the cache. Typical caching policies are described in terms of their replacement mechanism, where given a candidate entry, how does one decide whether to include it in the cache, and if so, which existing entry to evict in order to make space. Traditionally, the goal of a good policy yields a high *hit rate*, i.e. the fraction of times a query is in the cache. Meanwhile we distinguish that for CachingNets, a good policy will be one which maximizes the *trigger rate* of the classification module, namely the fraction of utterances which belong to the cache *and* the classification module identifies with high confidence *before* beam search concludes. While we use the hit rate as a motivator for our caching policies, it will serve only as an upper bound to the more relevant trigger rate. We propose and experiment with four different policies detailed below, each with their own merits.

3.1. Static Caching

In *static caching* we maintain a fixed set of N transcriptions in the cache at all times. No entries are evicted or added to the cache. The populated utterances are selected by the most frequently spoken sentences calculated globally across all users and time horizons. While static caching has a drawback in that it is not personalized to the user, it does have the advantage that since it is fixed, each training example references the exact same cache S. As a result, it avoids common "zero shot" learning problems and potentially helps the model identify hits and misses with greater precision.

3.2. Least Recently Used

Least recently used (LRU) is an eviction policy for the cache, where the entry which was least recently accessed is replaced by the new entry when the cache becomes full. Commonly implemented with some form of queue structure, for each user we maintain an individual queue of N transcripts. On a cache hit, we move that utterance to the back of the queue. On a cache miss, we add the new utterance to the back of the queue, while evicting the front-most entry which at this point will be the least recently spoken sentence. For our experiments, the cache is warmed (preset) for each user with the top global utterances and as a result, the cache maintains a constant size of N at all times. The policy is personalized, but lightweight, requires no external storage and exploits recency effects – recently spoken phrases have higher likelihoods of being repeated in the near future.

3.3. Least Frequently Used

When additional memory or services are available, more sophisticated caching strategies are possible. For instance, *least frequently used* (LFU) is a basic strategy where over some time horizon for each individual, we populate the cache with their most commonly spoken sentences and discard those which are accessed the fewest number of times. While LFU takes a more holistic, statistical approach to populating the cache which avoids using space for rare but recent sentences, LFU has the disadvantage of ignoring temporal aspects of good caching (e.g. 'play my christmas music playlist' is highly accessed but only relevant during one month of the year).

3.4. Ranking Based Approach

In order to bridge the gap between LRU and LFU policies, we also propose a more sophisticated model based caching scheme where we rank each user's past utterances by a propensity score representing the likelihood for near-future access. To accomplish this we build a model that combines both temporal features and semantic context based features. Training examples for our ranking model consist of pairs of a current utterance coupled with a list of past user utterance

[1] Note that there is also an opportunity to adjust a hypothesis score from the RNN-T beam if at the end of decoding the hypothesis happens to also be in S but was not triggered first. We do not report results on this here.

transcripts. The inputs to our ranking model is the concatenation of two vectors representing each modality of a user's unique past utterances. The temporal content is encoded as a d-dimensional counting vector where component d is the number of times the utterance was spoken d days in the past relative to the current date. For our experiments $d = 365$ representing a full year into the past. We use the 768 dimension BERT embeddings [26] to encode the semantic content of each past transcription. The temporal and BERT embeddings are then concetenated for each transcription under consideration.

We use a transformer-like encoder over the sequence of up to $K = 1000$ of a user's past utterances within the last year using frequency as a tie breaker. We feed these K vectors through a series of 4 transformer blocks of dimension 512, with four heads and 512 feed-forward hidden units. Note we do not apply positional coding before these transformer layers. The output for each of the K inputs is then projected with a dense layer to a single scalar and sent through a sigmoid activation. We use a binary cross entropy loss across all K propensity scores where the target class is 1 if and only if the current utterance is a match with that past transcription. At runtime we use the model to rank the K past unique utterances. We then populate the cache of the top N utterances with the highest propensities.

4. EXPERIMENTAL RESULTS

We investigate the model performance of the CachingNet architecture on a production VA ASR task. The training dataset consists of 100k hours of transcribed audio data of de-identified, far-field, multi-locale English (e.g. en-US, en-IN, etc.) utterances across common domains such as Music, Home Automation, and Notifications. The dataset also includes related historical transcriptions over a 12-month period which are used as input for the caching policy. For scaling purposes, we use a semi-supervised approach to gather historical transcriptions using a large production ASR system with second pass LM scoring, contextual labeling procedures and other post-ASR processing steps where rewrites are applied.

Audio features are sampled at 16 kHz and extracted using log-Filterbank Energies (LFBE) with 64 dimensions. Feature frames are downsampled by a factor of 3 and stacked with a stride size of 2 to produce each input frame. Our baseline model is a traditional RNN-T similar to the configurations of [17, 19] built with five LSTM encoder layers with 1024 units per layer, two LSTM decoder layers with 1024 units, and a single layer joint network with input projections of size 512. We use a vocabulary of 2.5k word-pieces generated by extracting the most frequent subword units (plus a blank symbol) and decode with a beam size of 16. We also build a baseline with FastEmit [27] regularization with constant 0.005 which very effectively encourages early emission of tokens. Our CachingNet models are built with the four policies of Static, LRU, LFU, and Ranking with a cache size of $N = 100$, critical threshold $\theta = 0.95$, and also include FastEmit regularization. The ranking model is trained first using the historical transcription data alone before using it to train the end-to-end CachingNet. Our models are trained until convergence using the Adam optimizer.

For each model, we record the WER, SER relative to the Baseline error rates.[2] For CachingNets we record the hit rate and trigger rate as described in Section 3. We also record the accuracy of the classification module for those utterances where it triggered. Finally, we measure the average EPL, the difference between when decoding completes and the last word is spoken by the user given by forced alignment. Table 1 summarizes these results. One observes that the CachingNet models show significant improvements in WER and especially in SER. One also sees that while caching policies can yield a wide hit rate range between 40% and 75%, the trigger ratio deviates far less. The static policy has the smallest gap between hit and trigger rates but the personalized caching trigger rates all hover near 50%. So even though the more sophisticated policies increase the hit rate, they do not necessarily correspond to a proportionally higher trigger rate. Furthermore, we observe that CachinNets are able to frontrun the RNN-T decoding and deliver significant latency gains, averaging over 75ms improvements. See Figure 2 for token emission sequences of an utterance compared to the caching classification probabilities. Table 2 shows the effects of altering θ and its impacts on predictive performance, trigger rates and EPL. Unsurprisingly, as θ increases, the trigger rates drop while EPL and accuracy increase with favorable operating points near 0.95 and 0.975.

[2]Due to internal company policy, we are not permitted to give absolute WER numbers, but we can state our baselines are below 7.5% WER and 15% SER absolute.

Model	WER (rel%)	SER (rel%)	Hits (%)	Trig. (%)	Acc. (%)	EPL (ms)
Baseline	–	–	NA	NA	NA	+69.6
FastEmit	-1.73	+0.68	NA	NA	NA	+14.5
Static	-3.46	-5.48	46.5	40.2	95.5	-63.97
LRU	-5.71	-6.16	69.9	51.3	96.7	-98.36
LFU	-7.37	-10.3	73.2	51.9	96.3	-97.81
Ranking	-7.44	-11.0	75.3	49.6	96.5	-85.26

Table 1. Predictive performance and latency measures for baseline transducer and CachingNet models. WER and SER are relative to the Baseline, and all other measures are in absolute terms.

θ	WER (rel%)	SER (rel%)	Hits (%)	Trig. (%)	Acc. (%)	EPL (ms)
0.80	3.46	-3.42	69.9	63.4	90.4	-146.1
0.90	-1.38	-5.48	69.9	56.9	93.9	-118.7
0.95	-5.71	-6.16	69.9	51.3	96.7	-98.36
0.975	-8.13	-7.53	69.9	46.3	97.3	-79.67
0.99	-7.44	-4.79	69.9	11.6	99.2	-1.186

Table 2. LRU CachingNet performance metrics with varying parameter θ.

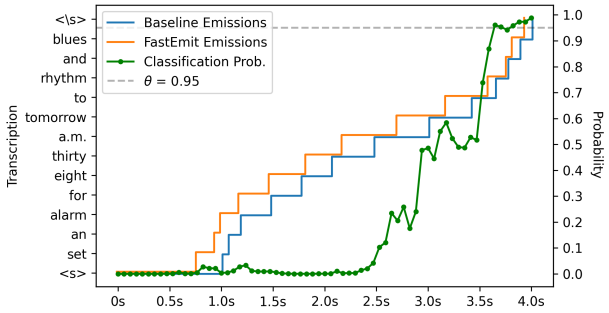

Fig. 2. Decoding emissions of an utterance for RNN-T 1-best baselines and CachingNet probability produced at each frame. See that the probability crosses θ before <\s> emission.

5. CONCLUSION

By applying different transcription caching policies, we show that the CachingNets architecture and its inference procedure can achieve significant WER and SER improvements over baselines with accompanying latency gains by leveraging user specific common utterances In future work, we would like to see new model-based policies designed to close the gap between cache hit and trigger rates and advanced strategies for choosing alternative cache capacities.

6. REFERENCES

[1] Alex Graves, "Sequence transduction with recurrent neural networks," *International Conference on Machine Learning (ICML)*, 2012.

[2] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals, "Listen, attend and spell," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964, 2016.

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

[4] Tara N. Sainath and Et al., "A streaming on-device end-to-end model surpassing server-side conventional model quality and latency," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6059–6063, 2020.

[5] Jinyu Li, Hu Hu, and Yifan Gong, "Improving RNN transducer modeling for end-to-end speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 114–121, 2019.

[6] Yanzhang He and Et al., "Streaming end-to-end speech recognition for mobile devices," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6381–6385, 2019.

[7] Jonathan Macoskey, Grant P. Strimel, Ariya Rastrow, and Alexa Machine Learning, "Learning a neural diff for speech models," *Proc. Interspeech*, pp. 2536–2540, 2021.

[8] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik Mcdermott, Stephen Koo, and Shankar Kumar, "Transformer transducer: a streamable speech recognition model with transformer encoders and RNN-T loss," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7829–7833, 2020.

[9] Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, Julian Chan, Frank Zhang, Duc Le, and Mike Seltzer, "Emformer: efficient nemory transformer based acoustic model for low latency streaming speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6783–6787, 2021.

[10] Anmol Gulati, James Qin, Chung Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang, "Conformer: convolution-augmented transformer for speech recognition," *Proc. Interspeech*, pp. 5036–5040, 2020.

[11] Suman Ravuri and Andreas Stolcke, "Recurrent neural network and LSTM models for lexical utterance classification," *Proc. Interspeech*, pp. 135–139, 2015.

[12] Kanthashree Mysore Sathyendra, Samridhi Choudhary, and Leah Nicolich-henkin, "Extreme model compression for on-device natural language understanding," *The 28th International Conference on Computational Linguistics (COLING)*, 2020.

[13] Ross Mcgowan, Jinru Su, Vince Dicocco, Thejaswi Muniyappa, Grant P. Strimel, and Alexa Machine Learning, "SmallER: scaling neural entity resolution for edge devices," *Proc. Interspeech*, pp. 761–765, 2021.

[14] Nishant Prateek and Others, "In other news: A bi-style text-to-speech model for synthesizing newscaster voice with limited data," *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pp. 205–213, 2019.

[15] Yuan Shangguan, Rohit Prabhavalkar, Hang Su, Jay Mahadeokar, Yangyang Shi, Jiatong Zhou, Chunyang Wu, Duc Le, Ozlem Kalinli, Christian Fuegen, and Michael L. Seltzer, "Dissecting user-perceived latency of on-device E2E speech recognition," *https://arxiv.org/abs/2104.02207*, 2021.

[16] Yuan Shangguan, Jian Li, Qiao Liang, Raziel Alvarez, and Ian McGraw, "Optimizing speech recognition for the edge," *arXiv preprint arXiv:1909.12408*, 2019.

[17] Jon Macoskey, Grant P Strimel, and Ariya Rastrow, "Bifocal neural asr : exploiting keyword spotting for inference optimization," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5999–6003, 2021.

[18] Yangyang Shi, Varun Nagaraja, Chunyang Wu, Jay Mahadeokar, Duc Le, Rohit Prabhavalkar, Alex Xiao, Ching-Feng Yeh, Julian Chan, Christian Fuegen, Ozlem Kalinli, and Michael L. Seltzer, "Dynamic encoder transducer: a flexible solution for trading off accuracy for latency," *https://arxiv.org/abs/2104.02176*, 2021.

[19] Jonathan Macoskey, Grant P. Strimel, Jinru Su, and Ariya Rastrow, "Amortized neural networks for low-latency speech recognition," *Proc. Interspeech*, pp. 4558–4562, 2021.

[20] Amazon.com, "Alexa Routines," *https://www.amazon.com/alexa-routines/*.

[21] Ian Williams, Anjuli Kannan, Petar Aleksic, David Rybach, and Tara N. Sainath, "Contextual speech recognition in end-to-end neural network systems using beam search," *Proc. Interspeech*, pp. 2227–2231, 2018.

[22] Golan Pundak, Tara N. Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: end-to-end contextual speech recognition," *IEEE Spoken Language Technology Workshop (SLT)*, pp. 418–425, 2019.

[23] Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf, "Contextual RNN-T for open domain ASR," *Proc. Interspeech*, pp. 1511–1515, 2020.

[24] Feng-Ju Chang, Jing Liu, Martin Radfar, Athanasios Mouchtaris, Maurizio Omologo, Ariya Rastrow, and Siegfried Kunzmann, "Context-aware transformer transducer for speech recognition," *To Appear IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2021.

[25] A. Vaswani and Others, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018.

[27] Jiahui Yu, Chung-Cheng Chiu, Bo Li, Shuo-yiin Chang, Tara N. Sainath, Yanzhang He, Arun Narayanan, Wei Han, Anmol Gulati, Yonghui Wu, and Ruoming Pang, "FastEmit: Low-latency streaming ASR with sequence-level emission regularization," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6004–6008, 2021.