GATED CONTEXTUAL ADAPTERS FOR SELECTIVE CONTEXTUAL BIASING IN NEURAL TRANSDUCERS

Anastasios Alexandridis, Kanthashree Mysore Sathyendra, Grant P. Strimel, Feng-Ju Chang, Ariya Rastrow, Nathan Susanj, Athanasios Mouchtaris

Amazon Alexa

ABSTRACT

Neural contextual biasing for end-to-end neural ASR transducers has shown significant improvements in the recognition of named entities, such as contact names or device names. However, it comes with the cost of increased compute, as the biasing layers (which are usually based on cross-attention) add complexity to the neural transducers. In this paper, we propose gated contextual biasing models that can estimate at runtime when contextual biasing is needed and can toggle it on or off. That way, contextual biasing does not run on every audio frame, but only on the frames where it can be helpful for correct ASR recognition. We show that our gated contextual biasing models can maintain all the performance improvements of contextual biasing while offering significant compute-cost saving, as the contextual biasing needs to be executed for fewer than 15% of the audio frames.

Index Terms— personalization, neural transducer, contextual biasing, end-to-end, contact name recognition

1. INTRODUCTION

End-to-end (E2E) ASR models are gaining increased interest due to their ability to outperform hybrid HMM-DNN systems, when sufficient training data is available [1]. They are usually based on neural transducers, such as the recurrent neural network transducer (RNN-T) [2], transformer [3] or conformer-transducer (C-T) [4, 5]. However, they still struggle to recognize personalized words that are uncommon and appear rarely in the training data, such as contact names, proper nouns, device names, and other named entities. Examples of such personalized interactions include utterances such as *"call [Contact Name]"* or *"turn on [Device Name]"*.

Prior work to integrate personalized context can be broadly categorized into two classes: post-training integration and in-training integration [6–12]. The first category is only applied during inference and includes integration of external language models, such as shallow-fusion [9,13] and on-the-fly re-scoring [14], which construct n-gram weighted finite state transducers (WFSTs) in order to boost contextual entities. The second class includes approaches that perform neural contextual biasing on the E2E ASR model [7, 15–17]. A popular approach is the contextual LAS model [7], which encodes contextual entities through a bias encoder and uses a location-aware attention mechanism to bias the ASR model towards them. Neural contextual biasing was later introduced for RNN-T and transformer transducers in [10, 15, 16]. These approaches train the neural transducer (augmented by a contextual biasing module) from scratch in order to boost contextual entities (via an attention mechanism) that are encoded using a catalog encoder. The approaches were later extended in [17], which proposed contextual adapters that are faster, cheaper to train and more data efficient through the use of a pre-trained neural transducer and an adaptation stage.

Neural contextual biasing has been shown to outperform post-training approaches like shallow-fusion and provide significant improvements on datasets containing named entities (henceforth referred to as "contextual entities"). The performance on generic speech datasets (that do not contain contextual entities) is maintained through the addition of a $< no\text{-}bias >$ token in the contextual entities, which teaches the neural adapter not to bias when no contextual entity is present in the utterance. However, the neural biasing mechanism increases the complexity of the neural transducer architecture. While false-biasing is avoided with the use of the $< no\text{-}bias >$ token, the biasing mechanism, which is based on scaled dot-product attention, has to run on every frame, increasing the computational complexity and the latency of the overall system. This is even more pronounced as the number of contextual entities increases, which requires the biasing mechanism to attend over several hundreds or even a few thousands of entities.

We propose training *gated* contextual adapters to reduce the compute cost of contextual biasing. This can be especially desirable for cases where the ASR model runs on devices with limited resources and compute capabilities. We extend the contextual adapters proposed in [17] by enabling them to toggle on and off based on whether a specific frame needs to be biased towards the contextual entities or not. This adds a form of dynamic compute, which has found use in other speech recognition applications [18–21]. We show that our gated contextual adapters can maintain all the accuracy improvements that come from contextual (non-gated) biasing while offering significant compute-cost savings as the adapter needs to be activated for less than 15% of the audio frames.

2. CONTEXTUAL NEURAL TRANSDUCERS

Neural transducer E2E ASR models typically consist of an encoder network, a prediction network and a joint network. The encoder consumes the T input audio frames $\mathbf{x_t} = (x_0, \cdots, x_t)$ and produces high-dimensional representations h_t^{enc}. The prediction network consumes previously predicted word-pieces $\mathbf{y_{u-1}} = (y_0, \cdots, y_{u-1})$ and outputs h_u^{pred}. The joint network first combines h_t^{enc} and h_u^{pred} via a joint operation and passes the output through a series of feed-forward layers with activations and a final *softmax* in order to produce the probability distribution over word-pieces. The encoder and decoder are typically stacked RNN layers [2], transformer [3] or conformer [4,5] blocks. The model is trained with the RNN-T loss using the forward-backward algorithm [2].

The contextual adapters, proposed in [17], augment the neural transducer by adding two components: an encoder for a catalog of contextual entities (catalog encoder) and a biasing adapter (shown in blue in Figure 1). The catalog encoder is responsible for embedding the list of contextual entities into encoded representations (hence-

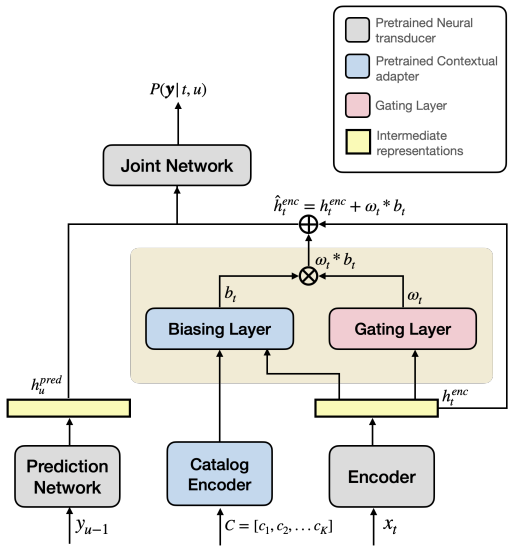


Fig. 1: Training architecture of the gated contextual adapters.

forth referred to as "entity embeddings"). The entities are tokenized into word-pieces and are passed through an embedding lookup and a series of BiLSTM layers. To avoid biasing when a contextual entity is not present, the list of entities is augmented by a special $< no\text{-}bias >$ token. For K catalog entities $C = [c_1, c_2, \cdots, c_K]$, the catalog encoder produces $C^e = [c_1^e, c_2^e, \cdots, c_K^e]$ entity embeddings. The biasing layer is responsible for adapting the neural transducer's intermediate representations based on the entity embeddings. This is done by a cross-attention mechanism to attend over the C^e entity embeddings based on the input query representation q (typically the encoder outputs h_t^{enc}). The attention score α_i is computed for each entity i using scaled dot-product attention. Based on the attention scores, the weighted sum of the value embeddings (obtained by a linear projection $\mathbf{W}_v$ of the entity embeddings) is computed as $b_t = \sum_i^K \alpha_i \mathbf{W}_v c_i^e$. The biasing vector b_t is used to update the intermediate representations of the neural transducer ASR model. This is done by linearly projecting it to the dimensions of the intermediate representation and adding it to the intermediate representation, resulting in the updated representation $\hat{h}_t^{enc}$

3. GATED CONTEXTUAL ADAPTERS

We augment the contextual adapters with a gating layer, as shown in Figure 1. The gating layer informs whether contextual biasing is needed for an audio frame. During inference, the gating layer can toggle contextual biasing on or off. The gating layer is lightweight and can be trained in an adaptation stage using a pretrained contextual neural transducer (grey and blue blocks in Figure 1), thus making it easy and cheap to train. Note that we use the encoder representation to perform biasing; however, our approach is applicable to any intermediate representation (decoder or joint network states).

3.1. Gating layer

The gating layer takes the intermediate representation h_t^{enc} and passes it through a feed-forward layer with *tanh* activation. The output is projected to a scalar through another feed-forward layer and a sigmoid activation is applied to get a weight, $\omega_t \in [0, 1]$:

$$z_t = tanh(\mathbf{W_1} h_t^{enc} + \mathbf{b_1}) \tag{1}$$

$$\omega_t = sigmoid(\mathbf{W_2} z_t + \mathbf{b_2}) \tag{2}$$

Here, $\mathbf{W_1}$, $\mathbf{W_2}$, $\mathbf{b_1}$, and $\mathbf{b_2}$ are learnable weights. The gating layer produces a scalar weight ω_t (gate), for every audio frame, that contains information about how big the contribution of the bias vector has to be in the neural transducer intermediate state.

3.2. Gated adapter training

During training, the output weight ω_t of the gating layer is multiplied by the bias vector b_t and the intermediate encoder representation is updated via element-wise additions as:

$$\hat{h}_t^{enc} = h_t^{enc} + \omega_t * b_t \tag{3}$$

This way the gating layer scales (or gates) the contribution of the contextual adapter to the intermediate neural representation.

By design, the gated contextual adapter is built by auxiliary training of the gating layer. The neural transducer and contextual adapter (shown in grey and blue in Figure 1) are initialized by a pretrained contextual neural transducer. The pretrained parameters are kept frozen, while the gating layer is initialized randomly and trained from scratch. Since the gating layer interacts with the biasing layer only with a multiplication with the scalar weight w_t, the pretrained contextual ASR model architecture can remain the same. Thus, it allows for a fast and cheap integration of the gated neural adapter, as only the gating layer needs to be trained for a small number of epochs.

3.3. Loss function

The gated contextual adapter is trained by enhancing the RNN-T loss with a regularization term. The goal is to teach the model that when no contextual biasing is needed, the weights that are output by the gating layer must be small. We investigate two different loss configurations:

$$\mathcal{L} = \mathcal{L}_{RNN-T} + \frac{\lambda}{T} \sum_{i=1}^{T} \omega_i \tag{4}$$

$$\mathcal{L} = \mathcal{L}_{RNN-T} + \frac{\lambda}{T} \sum_{i=1}^{T} \omega_i^2 \tag{5}$$

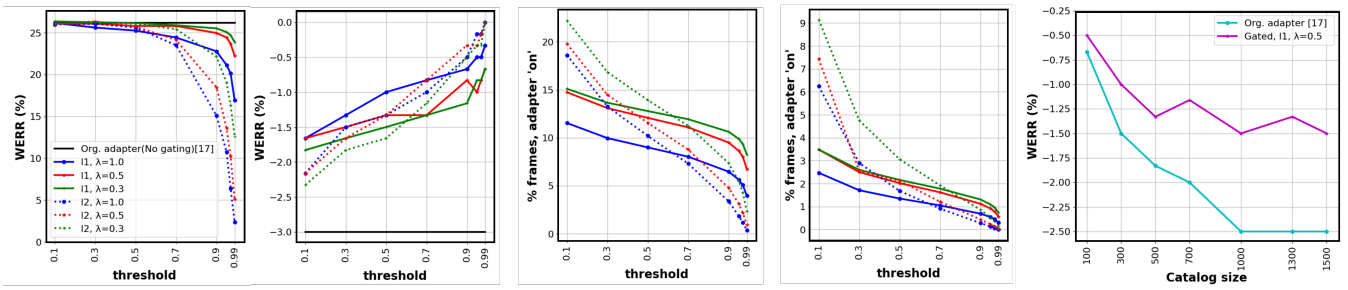
Eq. (4) performs ℓ_1-regularization to the weights of the gating layer, while Eq. (5) performs an ℓ_2-regularization. The loss penalizes high weight values and incentivizes the network to deactivate the biasing layer more often. The hyperparameter λ controls how aggressively the model prioritizes on outputting small weights over maintaining the accuracy of the contextual adapter.

3.4. Inference

During inference, the weight acts as a gate to toggle the biasing layer on and off. A hard-thresholding operation is applied to the weight and the neural transducer intermediate state is updated as follows:

$$\hat{h}_t^{enc} = \begin{cases} h_t^{enc}, & \text{if } \omega_t <= \epsilon \\ h_t^{enc} + b_t, & \text{otherwise} \end{cases} \tag{6}$$

The above rule implies that when the gating weight is less than a pre-defined threshold ϵ, the biasing layer can be switched off completely. Since the biasing layer is based on an attention mechanism in which the keys (i.e., entity embeddings) can be of a large dimension, this in turn saves compute cost (and thus latency).



(a) WERR, *named entities*

(b) WERR, *general*

(c) % frames with adapter enabled – *named entities*

(d) % frames with adapter enabled – *general*

(e) WERR, *general*, different catalog sizes

Fig. 2: Relative WER reduction (WERR) compared to a non-contextual neural transducer and % of frames where the contextual adapter is executed for different gated contextual adapter models.

4. EXPERIMENTAL SETUP

Datasets. We use in-house de-identified far-field datasets from voice assistant traffic, consisting of audio and corresponding transcriptions. The training set to train the baseline neural transducer model contains $\sim$ 114k hours of data. For training the neural adapters and the gating layers we use 290 hours of data that contains a mix of *specific* and *general* training data in a ratio of 0.4:1. Specific datasets contain utterances in the *Communication* domain with mentions of named entities (NE), e.g., proper names that can be used for biasing. *General* utterances are sampled from the original training data distribution. We evaluate our models and report results on two test sets: a *specific (NE)* test set of 22 hours and a *general* test set of 75 hours.

Evaluation metrics. As evaluation metrics, we report the relative word error rate reduction (WERR) on the two test sets. Given a model A's WER (WER$_A$) and a baseline model B's WER (WER$_B$), the WERR of A over B is WERR $=$ (WER$_B$ $-$ WER$_A$)/WER$_B$. Higher values indicate better performance. A negative value indicates degradation compared to the baseline. WERR is reported relative to the baseline neural transducer with no contextual biasing. For the gated contextual adapter models, we also report the fraction of frames where the biasing layer will need to be executed, as a proxy of the compute-cost reductions of gated contextual biasing.

Baseline neural transducer. We use a conformer-T [4, 5], which was trained on $\sim$ 114k hours of randomly sampled general voice assistant data. The input features are 64-dimensional Log-filter bank energies (LFBEs) extracted by segmenting utterances with a window of 32 ms and frame rate of 10 ms. We use a left context of 3 frames, resulting in 192-dimensional input feature vectors, with a skip rate of 3 frames. The features are normalized with global mean and variance. The conformer encoder network consists of two convolutional layers followed by 14 conformer blocks. The convolutional layers have 128 kernels of size=3 and strides=2 and 1 for the first and second convolutional layer, respectively. Each conformer block contains a feed-forward network module with 1024 units, a convolutional module with kernel size 15 and an attention module with 8 64-dimensional attention heads. All convolutions and attentions are computed on the current and previous audio frames to make the encoder streamable. The prediction network consists of 2 LSTM layers with 1024 units. The output of the encoder and decoder are projected to 512 units. The joint network consists of a feed-forward layer of 512 units. A 4k word-piece tokenizer [22] is used to create the output tokens. During decoding we perform the standard conformer-T beam search with a beam size set to 8. The total parameters of the model is 81M. For training, we used the Adam optimizer with varied learning rate following [3, 23].

Baseline contextual adapter. For the contextual adapters, the catalog encoder is a Bi-directional LSTM with 128 units with an input size of 64. The final output is projected to a 64-dimensional representation. The biasing layer projects the query, key, and values into 128 dimensions and the final biasing vector is projected to the same output as the encoder intermediate representation using a linear projection. For training the neural adapters we used the Adam optimizer with a learning rate of 1.2e-3. The models are trained with early stopping. During training the maximum catalog size (i.e., entities in the catalog encoder) is set to 100 in order to fit in memory.

Gated contextual adapter. For the gating layer we use a feed-forward layer with 128 units. The output is consumed by a second feed-forward layer that outputs the weight ω_t. We use the same *specific* and *general* training sets to train the gated adapter as we used for the baseline contextual adapter. We used an Adam optimizer with a learning rate of 1.2e-3. Note that the gating layer is trained on another adaptation stage using a pretrained contextual ASR (i.e., neural transducer + catalog encoder + biasing layer) model. Thus, the neural transducer is trained first and used as a pretrained model to train the catalog encoder and biasing layer, through an adaptation step. This model is then used as pretrained model (keeping its parameters frozen) in order to train the gating layer at a second adaptation step. The gating layer is comprised of only 65K parameters which is less than 0.1% of the total parameters of the model and less than 12% of the contextual adapter. During the second adaptation, the maximum catalog size is again set to 100. During inference, all the contextual adapters use a variable catalog size (based on the number of named entities available for each utterance), with a maximum catalog size of 5K. The mean catalog size is 1500 entities.

5. RESULTS

Figure 2 presents the results of different gated adapter models for different operating thresholds ϵ. The models were trained with different loss configurations (Eq. (4) and (5)) and various settings for the hyperparameter λ. It can be observed that all models can achieve similar WERR on *named entities* as the baseline contextual adapter with no gating (black line), for thresholds up to 0.5 (Figure 2a). After 0.5, the performance improvements start to deteriorate, with the models trained on ℓ_2 regularization deteriorating much faster. The reason is that as we become more aggressive in toggling the contextual adapter off, frames that benefit from contextual biasing are not biased anymore. In terms of ℓ_1 versus ℓ_2 regularization, it is ex-

Table 1: Ablation. Hard- vs. soft-thresholding during inference

	WERR (%)	
	Named Entities	General
Soft	26.23	-1.33
Hard, $\epsilon = 0.1$	26.05	-1.66
Hard, $\epsilon = 0.3$	26.05	-1.50

pected that ℓ_1 will have superior performance as it is popularly used to induce sparsity, i.e. push the gating weights close to zero. Regarding the WERR in the *general* dataset (Figure 2b), we observe a degradation of 3% for the original contextual adapter with no gating. This degradation is consistent with the results presented in [17] and shows a small degree of false-biasing. However, the gated contextual adapters are able to offer improved performance, reducing the degradation of the original contextual adapter with no gating. This shows that the model learns to toggle the adapter off when no context is present and avoids false-biasing.

Figure 2 also shows the fraction of time frames where the model chooses to perform biasing for the *specific* (Figure 2c) and *general* (Figure 2d) datasets. As expected, the fraction of frames where the adapter is enabled is decreased with increasing ϵ. Note that the fraction of frames to bias are much smaller in the *general* dataset, which validates that the gated adapter is learning to toggle on or off based on the presence of context. Comparing different values of λ, the tradeoff between performance and how aggressively the adapter is disabled is evident. Smaller values of λ are able to maintain performance better (especially for large threshold values) at the expense of running the contextual adapter on a larger number of frames.

Overall, the best performance is achieved for the models trained with the ℓ_1 regularization term and $\lambda = 0.5$ and 0.3. The models are able to maintain all the improvements gained by the contextual adapter with no gating and only run the biasing layer for less than 15% of frames for the *specific* dataset and less than 4% of the frames for the *general* dataset. This translates into significant compute cost savings, as for over 85% (*specific*) and 96% (*general*) of frames the contextual biasing can be switched off and recognition can be executed by the neural transducer alone. For all subsequent experiments we use the gated adapter model with ℓ_1 regularization and $\lambda = 0.5$.

Reducing false-biasing. Motivated by the results in Figure 2b, we further investigate the ability of the gated adapters to reduce false-biasing in general test sets. Figure 2e compares the performance of the original adapters [17] and the proposed gated adapter in the *general* test set for different catalog sizes. As observed in [17], there is a small degradation, that becomes more evident as the number of contextual entities becomes larger. However, the gated contextual adapters are able to reduce the degradation, offering improved performance. The reason is that the model toggles the adapter completely off when biasing is not needed, which avoids false-biasing.

Impact of hard-thresholding during inference. To evaluate the effects of gating with a hard-threshold on ω_t during inference, we evaluate the gated contextual adapter by running inference with a soft-threshold. Instead of performing the hard-thresholding operation in Eq. (6), we simply multiply the weight with the biasing vector as we do when training the gated adapter in Eq. (3). The results are shown in Table 1 which validates that hard-thresholding does not negatively impact the model for small thresholds: marginal differences are observed between the two inference modes.

Impact of regularization term. To evaluate the importance of the regularization term in the loss function, Table 2 compares the gated biasing adapter ($\ell_1, \lambda = 0.5$) to a gated biasing model we trained

Table 2: Ablation. Effect of regularization term in gated adapters

	WERR (%)		% frames, biasing "on"	
	Named Ent.	General	Named Ent.	General
$\epsilon = 0.1$	26.05	-1.66	14.74	3.48
w/o reg.	26.05	-2.83	99.9	99.88
$\epsilon = 0.99$	22.21	-0.67	6.73	0.56
w/o reg.	19.29	-2.00	64.52	66.34

(a) *"call harry potter"*

(b) *"call mom"*

Fig. 3: Visualization of the gate values

Table 3: Original contextual adapter gating using no-bias entity

	WERR (%)		% frames, biasing "on"	
	Named Ent.	General	Named Ent.	General
Org. CA (no gating)	26.14	-3.00	100	100
Org. CA (gating)	25.59	-1.0	35.70	6.21

just on the RNN-T loss (not including regularization). We can observe that for the same thresholds, the model without regularization achieves similar or worse performance, while selecting to bias almost all ($\epsilon = 0.1$) or more than 50% of the frames ($\epsilon = 0.99$).

Gating ability of original contextual adapters. We study the ability of the original contextual adapter [17] to perform gating, based on the value of the $< no\text{-}bias >$ entity which is present in the contextual entities. To do that, we do not perform biasing in the frames where the $< no\text{-}bias >$ entity has the maximum attention score over all entities. Table 3 shows that this form of gating can provide some level of discrimination between contextual and non-contextual frames; however, our proposed gated adapters outperform it by providing a lower number of frames where the adapter is activated. Note also, that this form of gating does not come with any compute-cost savings, since in order to estimate the no-bias score, we need to run the attention module, which is the biggest source of added complexity in the contextual adapters.

Visualizations. Finally, Figure 3 shows the gating values ($\epsilon = 0.1$) for every frame during decoding for two contextual utterances taken from the *specific* test set. It can be clearly seen how the gates are activated (white color) in the frames corresponding to the contextual entity "harry potter", which validates that our gated contextual adapters are learning to toggle the adapter on based on context. Also note that while "mom" is provided as a contextual entity, the token can be easily recognized by the transducer without biasing and the gate toggles the adapter off. This validates that the gated adapter is learning to toggle on and off, based both on context and confidence.

6. CONCLUSION

We proposed gated contextual biasing of E2E ASR neural transducers as a way to reduce compute cost of contextual ASR models. We trained a contextual adapter that can learn to toggle contextualization on or off based on the presence of context and showed that we can maintain all the performance improvements of contextual biasing by only performing biasing on less than 15% of the audio frames.

7. REFERENCES

[1] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4774–4778.

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

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

[4] 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," 2020.

[5] Bo Li, Anmol Gulati, Jiahui Yu, Tara N. Sainath, Chung-Cheng Chiu, Arun Narayanan, Shuo-Yiin Chang, Ruoming Pang, Yanzhang He, James Qin, Wei Han, Qiao Liang, Yu Zhang, Trevor Strohman, and Yonghui Wu, "A better and faster end-to-end model for streaming asr," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5634–5638.

[6] Tony Bruguier, Fuchun Peng, and Françoise Beaufays, "Learning personalized pronunciations for contact names recognition," 2016.

[7] Golan Pundak, Tara N Sainath, Rohit Prabhavalkar, Anjuli Kannan, and Ding Zhao, "Deep context: end-to-end contextual speech recognition," in *2018 IEEE spoken language technology workshop (SLT)*, 2018, pp. 418–425.

[8] Antoine Bruguier, Rohit Prabhavalkar, Golan Pundak, and Tara N Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6171–6175.

[9] Aditya Gourav, Linda Liu, Ankur Gandhe, Yile Gu, Guitang Lan, Xiangyang Huang, Shashank Kalmane, Gautam Tiwari, Denis Filimonov, Ariya Rastrow, et al., "Personalization strategies for end-to-end speech recognition systems," in *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7348–7352.

[10] Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf, "Contextual rnn-t for open domain asr," *arXiv preprint arXiv:2006.03411*, 2020.

[11] Duc Le, Gil Keren, Julian Chan, Jay Mahadeokar, Christian Fuegen, and Michael L Seltzer, "Deep shallow fusion for rnn-t personalization," in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 251–257.

[12] Duc Le, Mahaveer Jain, Gil Keren, Suyoun Kim, Yangyang Shi, Jay Mahadeokar, Julian Chan, Yuan Shangguan, Christian Fuegen, Ozlem Kalinli, et al., "Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion," *arXiv preprint arXiv:2104.02194*, 2021.

[13] Ding Zhao, Tara N. Sainath, David Rybach, Pat Rondon, Deepti Bhatia, Bo Li, and Ruoming Pang, "Shallow-Fusion End-to-End Contextual Biasing," in *Proc. Interspeech 2019*, 2019, pp. 1418–1422.

[14] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[15] Zhehuai Chen, Mahaveer Jain, Yongqiang Wang, Michael L Seltzer, and Christian Fuegen, "Joint grapheme and phoneme embeddings for contextual end-to-end asr.," in *Proc. Interspeech*, 2019, pp. 3490–3494.

[16] Feng-Ju Chang, Jing Liu, Martin Radfar, Athanasios Mouchtaris, Maurizio Omologo, Ariya Rastrow, and Siegfried Kunzmann, "Context-aware transformer transducer for speech recognition," *ASRU*, 2021.

[17] Kanthashree Mysore Sathyendra, Thejaswi Muniyappa, Feng-Ju Chang, Jing Liu, Jinru Su, Grant P. Strimel, Athanasios Mouchtaris, and Siegfried Kunzmann, "Contextual adapters for personalized speech recognition in neural transducers," in *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8537–8541.

[18] Jonathan Macoskey, Grant P. Strimel, and Ariya Rastrow, "Bifocal neural asr: Exploiting keyword spotting for inference optimization," in *2021 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021.

[19] Jonathan Macoskey, Grant P. Strimel, Jinru Su, and Ariya Rastrow, "Amortized neural networks for low-latency speech recognition," in *INTERSPEECH*, 2021.

[20] 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 Seltzer, "Dynamic encoder transducer: A flexible solution for trading off accuracy for latency," in *INTERSPEECH*, 2021, pp. 2042–2046.

[21] Yi Xie, Jonathan Macoskey, Martin Radfar, Feng-Ju Chang, Brian King, Ariya Rastrow, Athanasios Mouchtaris, and Grant Strimel, "Compute cost amortized transformer for streaming asr," in *Interspeech 2022*, 2022.

[22] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *ACL*, 2016.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NeurNIPS*, 2017.