Optimizing Contextual Speech Recognition Using Vector Quantization for Efficient Retrieval

Nikolaos Flemotomos[†], Roger Hsiao[†], Pawel Swietojanski[†], Takaaki Hori, Dogan Can, Xiaodan Zhuang
Apple
Cupertino, USA
{nflemotomos, rhsiao, pswietojanski}@apple.com

***Abstract*—Neural contextual biasing allows speech recognition models to leverage contextually relevant information, leading to improved transcription accuracy. However, the biasing mechanism is typically based on a cross-attention module between the audio and a catalogue of biasing entries, which means computational complexity can pose severe practical limitations on the size of the biasing catalogue and consequently on accuracy improvements. This work proposes an approximation to cross-attention scoring based on vector quantization and enables compute- and memory-efficient use of large biasing catalogues. We propose to use this technique jointly with a retrieval based contextual biasing approach. First, we use an efficient quantized retrieval module to shortlist biasing entries by grounding them on audio. Then we use retrieved entries for biasing. Since the proposed approach is agnostic to the biasing method, we investigate using full cross-attention, LLM prompting, and a combination of the two. We show that retrieval based shortlisting allows the system to efficiently leverage biasing catalogues of several thousands of entries, resulting in up to 71% relative error rate reduction in personal entity recognition. At the same time, the proposed approximation algorithm reduces compute time by 20% and memory usage by 85-95%, for lists of up to one million entries, when compared to standard dot-product cross-attention.**

***Index Terms*—contextual biasing, vector quantization, speech recognition, finite scalar quantization, retrieval**

I. Introduction

End-to-end deep learning models have revolutionized the field of automatic speech recognition (ASR), offering both simplicity and impressive performance [1]. However, even large ASR models trained on vast amounts of data still struggle to recognize rare words—in particular names of entities [2]–[4]. One way to address this problem is to allow models to leverage user-specific information, or context, during inference. Such context is often referred to as the user's biasing catalogue, and can include contact names, commonly used apps, media titles and creators, or relevant geo-locations.

To that end, several approaches have been proposed in the literature, including shallow language model (LM) fusion [5], [6] and on-the-fly LM-based rescoring [7], [8]. Such methods, though, are often sub-optimal, because of the well-documented tendency of end-to-end models to learn a strong internal LM [9], [10], and the fact that the contextual model is typically trained independently from the acoustic model (AM). Neural contextual biasing (NCB) offers an alternative paradigm where the biasing mechanism is part of the ASR model and jointly learned with the main ASR objective [11]–[18].

NCB architectures usually rely on an additional context encoder followed by a fusion mechanism. The context encoder may be implemented as an LSTM [19], or more recently as a transformer [20] module, and its role is to map a set of tokenized biasing phrases to a set of fixed-length continuous phrase embeddings that are integrated into ASR model predictions. This integration typically takes place in a latent space using cross-attention between audio and context encoders [11], [15]. Another approach relies on utilizing large language models (LLMs) for contextualization. In this case, an LLM is used jointly with the ASR model and the biasing entries are packed into a prompt [21] guiding LLM predictions [22]–[25].

Neural contextual biasing based on LLM prompting (thus on self-attention) or standard cross-attention fusion comes at the cost of increased compute and memory requirements, which means its applicability is limited to relatively small biasing catalogues in practical settings [11], [22], [26], [27]. On the one hand, LLMs have a fixed maximum context length, typically in the range of a few thousand tokens [21]. Even when maximum context length is not a blocker, LLM inference cost can be prohibitive for long contexts [21], and LLM reasoning performance can degrade rapidly as context length increases [28]. On the other hand, cross-attention, generally implemented as scaled dot-product [20], needs to be applied between all contextual embeddings (keys) and all acoustic encodings (queries), which can be restraining for large biasing catalogues. These limitations have led the research community to typically explore NCB settings where the biasing inventory is capped to a few thousand entries [11], [29]–[32].

In this work we address the attention-driven computational limitations of NCB employing a quantization-based, two-stage approach. We discretize contextual embeddings using finite scalar quantization (FSQ), a variant of vector quantization [33] recently introduced in generative machine learning architectures [34]. Utilizing certain properties of FSQ, we can approximate cross-attention in a memory- and runtime-efficient manner, and use this efficient implementation to accurately retrieve relevant biasing entries. These entries are then either (i) integrated with acoustic encodings through a dot-product cross-attention biasing mechanism, or (ii) used for LLM prompting in a delayed fusion setup [35].

Prior retrieval-based contextualization efforts typically either employ a pre-built entity index [36]–[38], allowing for

[†]equal contributions; alphabetical order

efficient search and retrieval, or are integrated within the ASR model [39], [40], allowing for frequent biasing list updates and joint optimization, but often requiring extensive compute and memory resources. Our proposed technique combines the best of the two worlds: it technically belongs to the latter category, but reduces the required memory budget through the introduced quantization and cross-attention approximation.

The main contributions of this paper are as follows:

1) We introduce vector quantization in the field of contextual speech recognition as a viable technique to discretize biasing embeddings and approximate the compute-heavy cross-attention mechanism.
2) We use our proposed technique to efficiently retrieve a short list of entries from a large biasing catalogue.
3) We pair our retrieval approach with different biasing implementations (full cross-attention, LLM prompting) and decoding configurations (auto-regressive, non auto-regressive), and demonstrate that our approach is flexible and agnostic to the specifics of the biasing mechanism or decoding algorithm.

Our experimental results show that our approximation algorithm can lead to over 20% speed boost and 85-95% memory usage reduction for biasing lists with at least 10k entries (Section V-C). At the same time, our retrieval approach can offer up to 71% relative error rate reduction for personal entity recognition, when the retrieved entities are jointly used for dense cross-attention and LLM prompting (Section V-B). Our work opens up a path to scaling neural contextualization to relatively unexplored scenarios such as biasing towards large media catalogues, where the number of entries could be in the hundreds of thousands or millions.

II. Background and Related Work

A. Cross-attention based contextualization

Typical transformer-based NCB models employ a cross-attention fusion layer where the encoded biasing phrases are integrated with the acoustic encodings [15], [17]. Although traditionally applied at the output of the audio encoder, some recent works report improved accuracy by doing fusion via one or more intermediate audio encoder layers [41]–[44]. Cross-attention fusion has also been extended to architectures containing additional branches and losses to detect the biasing phrases [32], [45], and copying mechanisms able to directly copy the relevant biasing entries to the output transcription [46], [47]. Other efforts have focused on building more robust NCB systems by employing alternative sampling strategies to construct the biasing lists during training [48], by applying hard negative mining [49], or by injecting additional text-only information from the same distribution as the contextual domain [18], [50].

For our own work, we build the proposed biasing system expanding on the CTC-AED model [51], where the biasing mechanism is implemented using cross-attention between acoustic encodings and the biasing inventory. Formally, $\boldsymbol{X} \in \mathbb{R}^{T\times D}$ denotes a sequence of acoustic embeddings, which are the output of a conformer-based acoustic encoder, and $\boldsymbol{C} \in \mathbb{R}^{|\mathbb{B}|\times D}$ denotes a list of contextual embeddings, as given by a transformer-based contextual encoder. Here D is the output dimension of the encoders, T is the length of the audio, and $\mathbb{B}$ is the list of biasing phrases, with $|\mathbb{B}|$ denoting its length.[1] Note that the biasing list, $\mathbb{B}$, always contains a special back-off token. The role of this special token is to help the model learn not to attend to a real biasing phrase whenever the utterance does not contain any contextual entity. Then, vanilla NCB computes

$$\{\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}\} = \{\boldsymbol{X}\boldsymbol{W}_q,\boldsymbol{C}\boldsymbol{W}_k,\boldsymbol{C}\boldsymbol{W}_v\} \tag{1}$$

$$\boldsymbol{Y} = \mathrm{softmax}\left(\alpha\, \boldsymbol{Q}\boldsymbol{K}^\top\right)\boldsymbol{V} \tag{2}$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are known as queries, keys, and values respectively; $\boldsymbol{Q}$ is computed by applying the linear transformation $\boldsymbol{W}_q \in \mathbb{R}^{D\times D}$ to $\boldsymbol{X}$; $\boldsymbol{K}$ and $\boldsymbol{V}$ are computed by applying the linear transformations $\boldsymbol{W}_k \in \mathbb{R}^{D\times D}$ and $\boldsymbol{W}_v \in \mathbb{R}^{D\times D}$, respectively, to $\boldsymbol{C}$; and $\alpha = 1/\sqrt{D}$ is a normalizing factor. The resulting biasing encodings, $\boldsymbol{Y}$, are added to the acoustic embeddings, $\boldsymbol{X}$, and fed into CTC and/or AED decoders, similarly to [41]. This process is referred to as *Dense NCB* and is visually depicted in Fig. 1a.

B. LLM based contextualization

LLMs are transformer-based models with billions of parameters that have yielded exciting results in general-purpose natural language understanding and generation [21]. The speech community has been exploring ways to incorporate the power of LLMs into ASR decoding [52]–[56], with shallow fusion [57], n-best rescoring [58], and error correction re-decoding [59] being some of the proposed approaches. Delayed fusion (DF) has also been recently introduced as an alternative [35], allowing one to use already trained LLMs and ASR models (potentially with very different vocabularies) to improve ASR quality without a large inference overhead.

LLM predictions can be further improved by injecting query-specific information via prompts [21]. Generally speaking, during prompting, a compatible model gets conditioned on some number of prefix tokens, either at the attention decoder [60] or encoder [42], [54] level. Several works have recently tried to take advantage of the prompting mechanism of LLMs as an alternative contextualization approach. For instance, [24] shows improved results transcribing technical talks when a small set of relevant keywords is included in the prompt, while [22] follows a similar approach, but reports a rapid degradation as the size of the biasing catalogue gets bigger. Prompting functionality has also been used in conjunction with retrieval mechanisms that operate at the textual [61], phonetic [25], or acoustically encoded sentence level [23], [38].

C. Towards efficient contextualization

Attention is a ubiquitous technology across the machine learning community [62] and significant effort has been spent on developing efficient attention implementations. For instance,

[1]The acoustic features which are given as input to the acoustic encoder are often downsampled; T represents the length of the downsampled sequence.

flash attention [63] introduces substantial speed gains by optimizing the read and write operations between different levels of memory, while ring attention [64] offers an elegant way to parallelize attention computation across multiple devices following block-wise processing. Such approaches, though, are hardware-oriented and do not address the fundamental memory requirements of attention computation.

In an effort to improve the computational efficiency of the biasing mechanism, some works dynamically limit the number of relevant phrases throughout decoding based on prefixes that appear in the partial hypotheses, applying bias-conditioning [11] or representing the biasing lists with prefix-trees [6], [16], [65]. Other works have proposed to constrain NCB to be only triggered on a subset of relevant audio frames, something that also helps with the phenomenon of over-biasing [26], [29]. Such approaches, despite limiting the number of time steps for which the dot-products need to be computed, still do not offer a scalable solution with respect to the length of the biasing list.

One way to handle a large biasing catalogue in a flexible manner is to follow a two-stage, retrieval-based approach. In that line of work, the authors in [18], [39], [66]–[68] apply a phrase-level attention to select a few candidate biasing phrases before applying a more expensive wordpiece-level attention [39], [68]. However, if N denotes the number of biasing phrases and k the number of wordpieces per phrase, this approach tries to find a trade-off between a latency of $O(N)$ and $O(kN)$, thus still struggles to scale well with the number of biasing entries, N.

Another two-stage method is introduced in [27], where the phone sequence of the streaming ASR output is used to filter out irrelevant entries of the biasing list. Phonemic information is also explored in [25], where a few candidate entities from a first pass are used to guide an LLM-based retrieval of phonologically similar entries and prompt the LLM during second-pass decoding. A closely related work is dual attention [69], a technique recently applied to keyword spotting [70]. The idea relies on having two attention modules, a small and a large (full) one, and conditionally executing one of them depending on triggering signals.

Retrieval approaches for ASR contextualization often operate in the linguistic [36] or acoustic embedding space [23], [37], where the embeddings act as queries to retrieve k nearest neighbours (kNN) [36] from a pre-built entity index, optionally using an auxiliary retrieval model [37], [40]. Recently, a joint approach has been explored, where the retrieval step is part of the contextual ASR model [39], [40]; in this case a form of dot-product score is still used to shortlist biasing entries, potentially employing an additional retrieval-oriented loss. The kNN and attention approaches are compared in [71], for the case of multi-modal LLMs. Index-based retrieval approaches allow for an efficient search, but introduce the cost of building and maintaining an up-to-date index, possibly aligned with auxiliary models that were used for indexing and/or retrieval. On the other side, joint model-level retrieval techniques are end-to-end learnable and self-contained, though at the higher cost of compute and memory requirements during inference. Our approach belongs to the latter category, though drastically limits the memory requirements during retrieval.

D. *Vector quantization*

Vector quantization (VQ) is a data compression technique [72], where a large set of points are represented by a finite smaller set of vectors, known as the codebook vectors. VQ codebooks can be automatically learned through gradient descent and used in conjunction with variational autoencoders to build robust generative models [33]. To that end, a VQ loss is added to the overall optimization objective, that essentially moves the codebook elements closer to the non-quantized points, based on their distances. Note that, instead of the learned quantized vectors, the original points can now be equivalently represented just by a single index in the learned codebook.

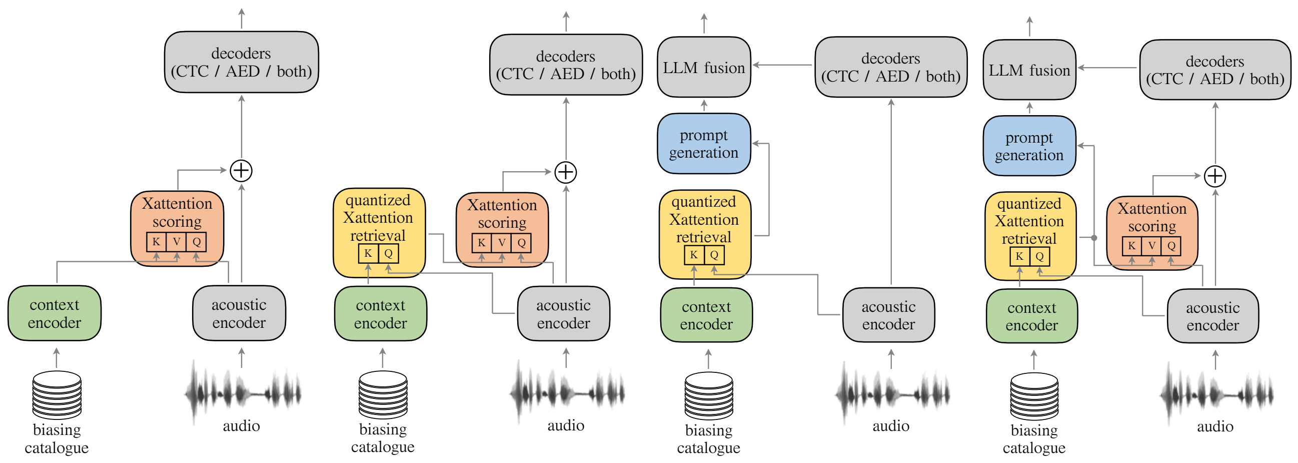
A multitude of improvements to the original VQ formulation have been proposed in the literature [73]–[77]. The most notable ones in the field of audio processing are residual vector quantization (RVQ) and grouped VQ. RVQ [74] employs a cascade of codebooks, where each one is responsible for the discretization of the quantization error, or residual, between the embedding and its quantized representation in the preceding layer, in a recursive manner. That way, the original points are represented by a set of indices (as many as the desired RVQ depth). Note that vanilla VQ would require exponentially larger codebooks to achieve the same quantization errors as RVQ. Grouped VQ [75] also aims at decreasing the required number of codebooks and codebook vectors, by splitting the original embeddings into groups and employing separately trained VQs to quantize each group.

Despite the remarkable results achieved with VQ and the aforementioned variants, training such discrete latent representations remains challenging. Codebook collapse, the model learning to only use a very small subset of the available codebook vectors, is a well-documented problem. Several solutions have been proposed, including alternative distance metrics, carefully designed initialization and learning rate updates, embedding estimation through exponential moving averages, and auxiliary losses [73], [78]–[80].

In an effort to simplify the original VQ formulation, thus overcoming the associated challenges, the authors in [34] introduced finite scalar quantization (FSQ) as an alternative. In FSQ, a d-dimensional vector (where d is relatively small, with the original embedding typically being projected onto a lower dimension) is quantized by mapping each one of its d values to an integer. To achieve that, the continuous values are first bounded through a non-linear function, and then rounded. Since there is a pre-defined set of levels $\mathbb{L} = [l_1, l_2, \cdots, l_d]$, and the i-th dimension can only take one of l_i different values (e.g., $\{-1, 0, 1\}$ for $l_i = 3$), the (implicit) codebook is of size $\prod_{i=1}^{d} l_i$. The idea of dropping explicitly trained codebooks is also explored in [81], where lookup-free quantization (LFQ) is introduced. LFQ operates similarly to a two-level FSQ, but training includes additional regularizations.

III. Proposed Architecture and Formulation

Although the Dense NCB formulation introduced in Section II-A stays relatively efficient for smaller biasing lists (say



(a) (b) (c) (d)

Fig. 1. Four considered biasing approaches: (a) vanilla contextual biasing referred to as *Dense NCB*; (b) *Retrieval NCB*, where a quantization module is used for large scale retrieval followed by TopK dense cross-attention processing; (c) *LLM Prompting*, where retrieved entries are packed into the prompt; (d) combination of *Retrieval NCB* and *LLM Prompting*. Note that in an efficient implementation, the context encoder needs to be run exactly once for each biasing entry and the encodings get cached and reused across multiple queries.

up to couple of thousand entries), the dot-product computation of Eq. (2) could consume much compute and memory, so it could become a bottleneck for larger catalogues. More precisely, we are computing $T \cdot |\mathbb{B}|$ D-dimensional dot-products. While both long audio (i.e., large T) and large inventory of biasing phrases (i.e., large $|\mathbb{B}|$) could cause problems, the audio is often less of a concern since it can be chunked to enable online recognition, or the biasing can be applied only to a subset of audio frames using gating [26]. The ability to handle a large inventory, $\mathbb{B}$, however, cannot be easily dealt with.

Here we propose a two-stage attention module, where the first efficient implementation (details in Section IV) is based on vector quantization and is used for retrieving biasing entries grounded on the audio queries. The full attention is still used for biasing only with those relevant entries, as depicted in Fig. 1b. Alternatively, the retrieved biasing phrases can be used for prompting an LLM that contextualizes the final ASR predictions through delayed fusion, as shown in Fig. 1c. The two approaches can also be combined, as in Fig. 1d.

Retrieval operates at the frame level, selecting the top entries from the dot-product between the queries and the keys. Such a frame-synchronous mode allows—depending on the target decoder—for both streaming and non-streaming applications.[2] This differentiates our systems from approaches operating in the hypothesis space, retrieving additional contextual signals for second-pass decoding only [82], or systems assuming sentence-level representation is available for retrieval [23]. While the ideas presented here could in principle be applied for sentence-level retrieval as well (e.g., by pooling all frame encodings into a sentence-level embedding and then retrieving the most relevant entries), a detailed comparison between frame-level and utterance-level retrieval is out of the scope of the current paper.

Note that during the retrieval stage we do not need to compute the values, $\boldsymbol{V}$, nor to calculate the softmax function in Eq. (2). While this further reduces the overall computational cost that would be otherwise required (in case of full cross-attention between audio queries and all the entries in the biasing list), the key novelty of our approach comes from the way we use vector quantization, and specifically FSQ, as detailed in the remainder of this section, and in Section IV.

In this work we apply FSQ to quantize the contextual embeddings of the NCB framework. In more detail, assume a contextual phrase with corresponding contextual embedding $\boldsymbol{c} \in \mathbb{R}^D$, as output from the NCB contextual encoder. Using FSQ with levels $\mathbb{L} = \left[l_1, l_2, \ldots, l_{|\mathbb{L}|}\right]$, we get the quantized representation, $\boldsymbol{z} = \mathrm{FSQ}(\boldsymbol{c})$. However, depending on the exact FSQ setting, the resulting codebook capacity might be very limited. For instance, for $\mathbb{L} = [8, 5, 5, 5]$, the implicit codebook size is equal to $\prod_i l_i = 1000$, whereas we want to represent hundreds of thousands or millions of potential biasing phrases. To increase the capacity of the quantizer, given a specific level configuration, $\mathbb{L}$, we apply grouping and we quantize separately each one of the groups. The initial vector, $\boldsymbol{c}$, is split into G groups and we quantize each one of the $\boldsymbol{c}^g \in \mathbb{R}^{(D/G)}$ sub-vectors. In our implementation, we use the same level configuration, $\mathbb{L}$, for the quantizers applied to all the sub-groups (for all $g \in \{1, 2, \ldots, G\}$). We get the final representation, $\boldsymbol{z}$, as follows:

$$\tilde{\boldsymbol{c}}^g = \boldsymbol{A}_{\text{in}}^g \boldsymbol{c}^g + \boldsymbol{b}_{\text{in}}^g = \left[\tilde{c}_1^g, \tilde{c}_2^g, \ldots, \tilde{c}_{|\mathbb{L}|}^g\right] \qquad \forall g \qquad (3)$$

$$\boldsymbol{e}^g = \left[\mathrm{round}[f_1(\tilde{c}_1^g)], \ldots, \mathrm{round}[f_{|\mathbb{L}|}(\tilde{c}_{|\mathbb{L}|}^g)]\right] \qquad \forall g \qquad (4)$$

$$\boldsymbol{n}^g = \mathrm{normalize}\left(\boldsymbol{e}^g\right) \qquad \forall g \qquad (5)$$

$$\boldsymbol{z}^g = \boldsymbol{A}_{\text{out}}^g \boldsymbol{n}^g + \boldsymbol{b}_{\text{out}}^g \qquad \forall g \qquad (6)$$

$$\boldsymbol{z} = \left[\boldsymbol{z}^{1\top}; \boldsymbol{z}^{2\top}; \ldots; \boldsymbol{z}^{G\top}\right]^\top \qquad (7)$$

where $[\boldsymbol{A}_{\text{in}}^g; \boldsymbol{b}_{\text{in}}^g]$ is the input affine transformation that projects the $\left({}^D/_G\right)$-dimensional vector $\boldsymbol{c}^g$ onto an $|\mathbb{L}|$-dimensional space; $[\boldsymbol{A}_{\text{out}}^g; \boldsymbol{b}_{\text{out}}^g]$ is the output affine transformation that projects the $|\mathbb{L}|$-dimensional vector $\boldsymbol{n}^g$ back onto the $\left({}^D/_G\right)$-dimensional space; $\mathrm{normalize}(\cdot)$ is a function that

[2]While for our experiments we stack all the frame-level retrieved entries into a single set which we employ during the second-stage biasing, a system deployed in a streaming scenario could do so in a chunk-wise fashion.

maps the integer elements of $\boldsymbol{e}^g$ to $[-1,1]$; $f_i(\cdot)$ is a bounding non-linear function such that each element e_i^g of $\boldsymbol{e}^g$ can take one of l_i different integer values. We use the function proposed in [34], $f_i(x) = \lfloor {}^{l_i}/{}_2 \rfloor \tanh(x)$. The embedding, $\boldsymbol{c}$, can be now represented by the G indices to the implicit codebooks.

During training, we load the parameters of a pre-trained, non-quantized NCB model and we only train the additional FSQ parameters (i.e., the projections $[\boldsymbol{A}_{\text{in}}^g; \boldsymbol{b}_{\text{in}}^g]$ and $[\boldsymbol{A}_{\text{out}}^g; \boldsymbol{b}_{\text{out}}^g]\ \forall g$), keeping the rest of the weights frozen.[3] For back-propagation to work through the non-differentiable rounding operation, we copy gradients via straight-through estimation (STE) [83].[4] Once the FSQ parameters have been trained, all the biasing phrases can be represented via the implicit FSQ codebooks, without the need to re-train any parameters based on the biasing catalogues at inference. Thus, the same model, without further tuning, can be used for various users, associated to different biasing lists, or to the same user after updating their own relevant biasing list (e.g., deleting or adding new contacts).

As already mentioned, each element e_i^g of $\boldsymbol{e}^g$ can take one of l_i different integer values for all the groups g; by extension, each element n_i^g of $\boldsymbol{n}^g$ can also take one of l_i different values. Denoting as $\mathbb{U} = \{u_j\}_{j=1..|\mathbb{U}|}$ the set of all possible values of $n_i^g\ \forall i \forall g$, we can easily see that, without loss of generality for the chosen bounding functions, the cardinality of the set is $|\mathbb{U}| \leq \sum_{i=1}^{|\mathbb{L}|} l_i$.[5] With that in mind, we can represent each contextual embedding, $\boldsymbol{c}$, by $G \cdot |\mathbb{L}|$ values from the set $\mathbb{U}$.

IV. VQ Based Algorithm for Cross Attention

A. Efficient dot-product estimation

As explained in Section III, we propose to apply FSQ to the contextual embeddings of the employed NCB architecture. This changes the structure of the dot-product appearing in Eq. (2). For simplicity, the formulation below assumes the number of groups is one, $G = 1$, however, we found that grouping is important when it comes to the representational power of FSQ.[6] Using a similar notation as in Eq. (6)—dropping the grouping indices—the dot-product between a query $\boldsymbol{q}$ from $\boldsymbol{Q}$ and a key $\boldsymbol{k}$ from $\boldsymbol{K}$, can be computed by

$$\begin{aligned}\boldsymbol{q}^\top \boldsymbol{k} &\simeq \boldsymbol{q}^\top \boldsymbol{W}_k \boldsymbol{z} \\ &= \boldsymbol{q}^\top \left[\boldsymbol{W}_k(\boldsymbol{A}_{\text{out}}\boldsymbol{n} + \boldsymbol{b}_{\text{out}})\right] \\ &= \boldsymbol{q}^\top \left[\boldsymbol{A}\boldsymbol{n} + \boldsymbol{b}\right] \\ &= \boldsymbol{q}^\top \sum_{i=1}^{|\mathbb{L}|} \boldsymbol{a}_i n_i + \boldsymbol{q}^\top \boldsymbol{b} \\ &= \sum_{i=1}^{|\mathbb{L}|} \boldsymbol{q}^\top \boldsymbol{a}_i n_i + \boldsymbol{q}^\top \boldsymbol{b} \qquad (8)\end{aligned}$$

where $\boldsymbol{z}$ is the quantized representation of the contextual embedding corresponding to $\boldsymbol{k}$; $\boldsymbol{A}$ and $\boldsymbol{b}$ are equal to $\boldsymbol{W}_k\boldsymbol{A}_{\text{out}}$ and $\boldsymbol{W}_k\boldsymbol{b}_{\text{out}}$, respectively; $\boldsymbol{a}_i$ is the i-th column of $\boldsymbol{A}$, i.e., $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{|\mathbb{L}|}]$; n_i is the i-th element of the $|\mathbb{L}|$-dimensional FSQ vector representation $\boldsymbol{n}$. By examining the above equations closely, we notice:

1) $\boldsymbol{A} \in \mathbb{R}^{D \times |\mathbb{L}|}$ and $|\mathbb{L}|$ is relatively small,
2) n_i can take l_i different values from $\mathbb{U}$,
3) $\boldsymbol{q}^\top \boldsymbol{b}$ is not needed for retrieval.

Properties 1 and 2 have a big impact on computation and memory consumption because we no longer compute the dot-product matrix $\boldsymbol{Q}\boldsymbol{K}^\top$ between the queries and a big inventory of keys as in Eq. (2). Remember that $\boldsymbol{Q}\boldsymbol{K}^\top$ is a matrix in $\mathbb{R}^{T \times |\mathbb{B}|}$ and its estimation requires computing $T \cdot |\mathbb{B}|$ D-dimensional dot-products between T different queries ($\boldsymbol{q}$) and $|\mathbb{B}|$ different keys ($\boldsymbol{k}$). Instead of those $|\mathbb{B}|$ different vectors, due to property 1, we now only have $|\mathbb{L}|$ vectors ($\{\boldsymbol{a}_i\}_{i=1..|\mathbb{L}|}$ in the notation of Eq. (8)), where $|\mathbb{L}|$ is often in the range $4-6$. Property 2 means there is limited variation for the values of n_i, since $|\mathbb{U}| \leq \sum_{i=1}^{|\mathbb{L}|} l_i$.[7] Combining properties 1 and 2 allows us to efficiently compute dot-products between the queries in $\boldsymbol{Q}$ and all possible vectors from $\boldsymbol{A}\boldsymbol{n}$. Since we have $|\mathbb{L}|$ columns in $\boldsymbol{A}$ and each element in $\boldsymbol{n}$ can take one of $|\mathbb{U}|$ different values, we get at most $|\mathbb{L}| \cdot |\mathbb{U}|$ D-dimensional vectors from $\boldsymbol{A}\boldsymbol{n}$. This would greatly reduce the memory consumption since $|\mathbb{B}| \gg |\mathbb{L}| \cdot |\mathbb{U}|$ for typical applications.

Property 3 implies we could ignore $\boldsymbol{q}^\top \boldsymbol{b}$ altogether since for retrieval this term only adds a different constant to each time step and does not affect the TopK selection that takes place at the frame level. Therefore, we pre-compute the score matrix $\boldsymbol{S}_{an} \in \mathbb{R}^{D \times (|\mathbb{L}| \cdot |\mathbb{U}|)}$:

$$\boldsymbol{S}_{an} = \left[\boldsymbol{a}_1 u_1, \boldsymbol{a}_1 u_2, \ldots, \boldsymbol{a}_{|\mathbb{L}|} u_{|\mathbb{U}|}\right] = [\ldots, \boldsymbol{a}_i u_j, \ldots] \qquad (9)$$

for all $i \in \{1, 2, \ldots, |\mathbb{L}|\}$ and $j \in \{1, 2, \ldots, |\mathbb{U}|\}$. To estimate the dot-product with a query, $\boldsymbol{q}$, we can compute

$$\boldsymbol{s}_{qan} = \boldsymbol{S}_{an}^\top \boldsymbol{q} \qquad (10)$$

where $\boldsymbol{s}_{qan} \in \mathbb{R}^{|\mathbb{L}| \cdot |\mathbb{U}|}$. Then, we can select the scores based on the $|\mathbb{L}|$ indices for a given biasing phrase, and sum them to get the desired dot-product between $\boldsymbol{q}$ and $\boldsymbol{W}_k\boldsymbol{z}$, which approximates $\boldsymbol{q}^\top \boldsymbol{k}$.

In sum, we use Eq. (10) to compute all possible dot-products between queries (acoustic encoder frames) and the keys (quantized contextual embeddings after FSQ). Then, we perform index selection and sum reduction to approximate $\boldsymbol{Q}\boldsymbol{K}^\top$ in Eq. (2). Finally, for each time frame we can retrieve the phrases with the highest dot-product scores. Algorithm 1 describes this process in pseudo-code, where $\boldsymbol{E}$ denotes the matrix that stores all the integer FSQ values corresponding to all the biasing phrases (the results of Eq. (4) for $G = 1$). Note that, once we have pre-calculated the score matrix $\boldsymbol{S}_{an}$, in practice we can run the algorithm in batches, with respect to both the time dimension (`for t = 1 to T`) and the biasing list (`for j = 1 to |B|`).

[3]Initial experimentation showed no improvements by training or fine-tuning all the network parameters jointly with the quantizer.

[4]The rounding operation of a tensor, `x`, can be implemented in PyTorch with STE as `x + (torch.round(x) - x).detach()`.

[5]We use the inequality sign to allow for the (common) scenario where the same values are used across multiple levels.

[6]In our implementation, for $G > 1$, we restricted $\boldsymbol{W}_k$ to be a block-diagonal matrix with G blocks, each one in $\mathbb{R}^{(D/G) \times (D/G)}$.

[7]For the settings used in our experiments, $\mathbb{L} = [8, 5, 5, 5]$ or $\mathbb{L} = [7, 5, 5, 5, 5]$, so that $|\mathbb{U}| \leq 27$.

Algorithm 1 FSQ-based dot-product computation and retrieval

```
Require: $\boldsymbol{A} \leftarrow \boldsymbol{W}_k \boldsymbol{A}_{\text{out}}$
Require: $\mathbb{L} \leftarrow$ FSQ level information
Require: $\mathbb{U} \leftarrow$ all possible values in FSQ codebooks
Require: $\boldsymbol{S}_{an} \leftarrow [\boldsymbol{a}_1 u_1, \ldots, \boldsymbol{a}_{|\mathbb{L}|} u_{|\mathbb{U}|}]$
  function TopK($\boldsymbol{X}$: Mat[$T$,$D$], $\boldsymbol{E}$: Mat[$|\mathbb{B}|$,$|\mathbb{L}|$], $K$: int)
    $\boldsymbol{Q} \leftarrow \boldsymbol{X}\boldsymbol{W}_q$
    $\boldsymbol{R} \leftarrow$ Mat[$T$,$K$]
    for $t = 1$ to $T$ do
      $\boldsymbol{s}^t_{qan} \leftarrow \boldsymbol{Q}[t,:]^\top \boldsymbol{S}^\top_{an}$
      $\boldsymbol{C} \leftarrow$ Zeros($|\mathbb{B}|$)
      for $j = 1$ to $|\mathbb{B}|$ do
        for $l = 1$ to $|\mathbb{L}|$ do
          $e_{jl} \leftarrow \boldsymbol{E}[j, l]$
          $\boldsymbol{C}[j] \leftarrow \boldsymbol{C}[j] + \boldsymbol{s}^t_{qan}[l \cdot |\mathbb{U}| + e_{jl}]$
        end for
      end for
      $\boldsymbol{R}[t] \leftarrow$ the indices of top $K$ values from $\boldsymbol{C}$
    end for
    return $\boldsymbol{R}$
  end function
```

Since we are interested in the retrieval scenario, in this section we focused on the quantization of the keys and the efficient dot-product estimation between keys and queries. Of course, since FSQ is applied on the contextual embeddings, and both keys and values are linear projections of those same embeddings, quantization of the values is also possible for a full quantized cross-attention.

B. Asymptotic runtime and space complexity

Table I summarizes the time and space complexity of the direct dot-product and our proposed approaches, when the size of the biasing list is the dominant factor, i.e., $|\mathbb{B}| \gg T, |\mathbb{U}|, |\mathbb{L}|, D, G$. The space complexity analysis would need to consider storage of queries, keys, and the output. The queries are the acoustic encodings and require $O(TD)$ space, which can be ignored when $|\mathbb{B}|$ is dominant. The output would require $O(T|\mathbb{B}|)$ space, if we wanted to store all the dot-products between the acoustic encodings and the biasing phrases. This is required in case of full cross-attention, in order to calculate the softmax. However, in case of retrieval, we can calculate and store dot-products in batches and only keep the TopK at every step, so we do not consider this for our analysis.

TABLE I
TIME & SPACE COMPLEXITY FOR DOT-PRODUCT COMPUTATION BETWEEN QUERIES AND KEYS WITH DIRECT AND PROPOSED APPROACHES.

Algorithm	Runtime	Space
Direct dot-product	$O(T\|\mathbb{B}\|D)$	$O(\|\mathbb{B}\|D)$
Quantized dot-product	$O(T\|\mathbb{B}\|\|\mathbb{L}\|G)$	$O(\|\mathbb{B}\|\|\mathbb{L}\|G)$

The size of biasing list, $|\mathbb{B}|$, is assumed as dominant factor.

Our proposed algorithm reduces memory consumption for the keys because we no longer store a D-dimensional vector for every biasing phrase. Instead, each phrase is represented by $|\mathbb{L}| \cdot G$ indices, so the space to store the keys becomes $O(|\mathbb{B}||\mathbb{L}|G)$. We use these indices to retrieve the scores from the matrix $\boldsymbol{S}_{an}$ in Eq. (9), which requires $O(|\mathbb{L}||\mathbb{U}|D)$ space, and can be ignored when $|\mathbb{B}|$ is dominant.[8] As a result, our proposed algorithm could save memory when $D > |\mathbb{L}| \cdot G$.

In practice, memory savings will be even more pronounced since we only need 2 – 3 bits to represent each of the $|\mathbb{L}|$ elements in the integer FSQ representation $\boldsymbol{e}^g\ \forall g$ (Eq. (4)). Therefore, it is possible to represent a biasing phrase with G 16-bit integers, i.e., $G \times 2$ bytes. However, for the vanilla dot-product approach, we would still need D floating point numbers for each biasing phrase, which is significantly larger even with low precision representation. This advantage of our proposed algorithm comes from the indexing structure that allows for a more compact representation.

To estimate the runtime complexity of our proposed algorithm, note that the score matrix $\boldsymbol{S}_{an}$ can be pre-computed before inference starts. Therefore, we only need to consider the dot-product computation between the queries and $\boldsymbol{S}_{an}$, which takes $O(T|\mathbb{L}||\mathbb{U}|D)$ time. Then, the index selection part would take $O(T|\mathbb{B}||\mathbb{L}|G)$ time (see Algorithm 1 for $G = 1$), which is the dominant factor when $|\mathbb{B}|$ is dominant. Therefore, the runtime complexity of our proposed algorithm would be lower when $D > |\mathbb{L}| \cdot G$, similarly to space complexity.

V. Experiments

We carry out the experiments on a large-scale in-house dataset consisting of examples from two tasks; dictation and assistant. Following [84], the model parameters are first estimated on semi-supervised data for a total of 500k updates, and then the contextual model is fine-tuned for another 100k updates on supervised data. In both stages, gradients are accumulated over 6,144 examples. We use SyncSGD + Adam [85] for distributed optimization, with exponentially decaying learning rates. The semi-supervised portion of the data consists of around 600,000 hours of automatically transcribed audio, while the supervised portion comprises about 50,000 hours of human-graded English queries; all data are anonymized at the user level.[9] For the quantized models, we freeze all the parameters and we only train the quantizers for another 100k updates. When training contextual models, similarly to [49], we sample 3 biasing phrases for each utterance (one positive and two distractors), then we share contextual phrases across all other utterances within a batch.

The backbone of our system is a CTC-AED model [51]. The acoustic encoder is a network with a Conv2D module, resulting in 6-fold downsampling of the input, followed by 12 conformer blocks. The AED decoder is a 3-block bi-directional transformer.[10] Both the encoder and decoder blocks have a hidden dimension of 2,048 and employ an 8-head self-attention. The context encoder comprises 3 transformer

[8]Note that the total space needed for the score matrix is not affected by the number of groups, G, since each group operates on a sub-vector in $\mathbb{R}^{(D/G)}$.

[9]Users are assigned code IDs, and it is not possible to map a code ID back to a real user.

[10]In practice, this is implemented with 3 blocks of forward transformers and 3 blocks of backward transformers. While all 6 blocks are used during training, the backward transformers are disabled during inference.

blocks with a hidden dimension of 512 and an 8-head self-attention. The biasing module is based on single-head cross-attention, since initial experimentation showed no benefits by employing multi-head attention. The input audio is enhanced by spectral augmentation [86] and is represented by 80-dimensional logmel features, extracted every 10msec. The text representation is based on a SentencePiece tokenizer [87] with a vocabulary size of 6k, trained on the transcripts of the supervised portion of our data. For LM-based experimentation, we employ an internal transformer-based neural network language model (NNLM) using the same 6k tokenizer, trained on the same transcribed data, as well as the public LLM OpenLLaMA 3B v2 [88], [89] with a vocabulary size of 32k.

Models are evaluated using an in-house test set containing 42 hours of data with queries targeted to a smart voice assistant. Around 50% of the test set comprises contextual queries containing contact, app, and media names, anonymized at the user level. The remainder portion of test data consists of examples that are generic in nature and are not expected to benefit from auxiliary biasing information. We report ASR results using two metrics, word error rate (WER [%]) for quantifying an average system performance, and named entity error rate (NEER [%]) to better capture contextualization performance. We further distinguish between NEER on contact names—that comprise the biggest portion of the biasing catalogues in our test set—and on all other (non-contact) entities. Note that NEER is a binary metric getting a score of 0 only when ASR correctly predicts the complete entity filler and 100 otherwise. While non-binary metrics, such as the biased word error rate (B-WER), have been also proposed in the literature [65], NEER is more appropriate for our use case where it is often imperative that the voice assistant perfectly recognize a named entity (e.g., when the user tries to make a call).

To showcase the flexibility of our systems with respect to the decoder applied, we perform experiments in both non auto-regressive and auto-regressive fashion. For the former, we apply CTC beam search followed by attention rescoring [90], the latter relies on joint CTC-attention decoding [91]. Both cases use a beam size of 10 and a relative CTC weight of 0.3.

A. Retrieval performance evaluation

In order to use the (quantized) NCB model to ground biasing catalogues on audio, we need to confirm that the attention head of the contextual cross-attention attends indeed to the expected entries. To do so, we feed to the NCB model an utterance with T frames that contains the contextual phrase $\hat{p}$ in the reference text, together with the corresponding biasing list $\mathbb{B}$ (with $\hat{p} \in \mathbb{B}$). At every frame t of the utterance we take the attention scores and we get the K phrases from the biasing list with the TopK scores (Fig. 1). Let's denote this set of phrases as $\mathbb{P}_K^t = \{p_i^t\}_{i=\{1..K\}}$. Note that we treat the back-off token as an empty phrase, so if the Top1 for frame t' is the back-off, then $\mathbb{P}_1^{t'} = \emptyset$. We then combine all the extracted phrases, for all the frames, into a set $\mathbb{S}_K = \mathbb{P}_K^1 \cup \mathbb{P}_K^2 \cup \cdots \cup \mathbb{P}_K^T$, and we calculate the *success rate* (or recall) as the rate for which $\hat{p} \in \mathbb{S}_K$ across our dataset.

We evaluate the retrieval capabilities of the models on a random subset of 1.4k examples drawn from the test set, all of which contain some contextual phrase (contact, app) anonymized at the user level, stored in the user's biasing list. Each example is uttered by a different speaker (for a total of 1.4k unique speakers), each one associated with their own biasing list. For this experiment we limit the maximum length of biasing catalogues to $\max|\mathbb{B}| = 5$k, randomly discarding biasing phrases if the initial user's catalogue is larger. We report the success rates for various FSQ configurations[11] in Fig. 2, while in Fig. 3 we report the number of retrieved phrases per utterance (i.e., the cardinality of the set $\mathbb{S}_K$ per our previous notation).

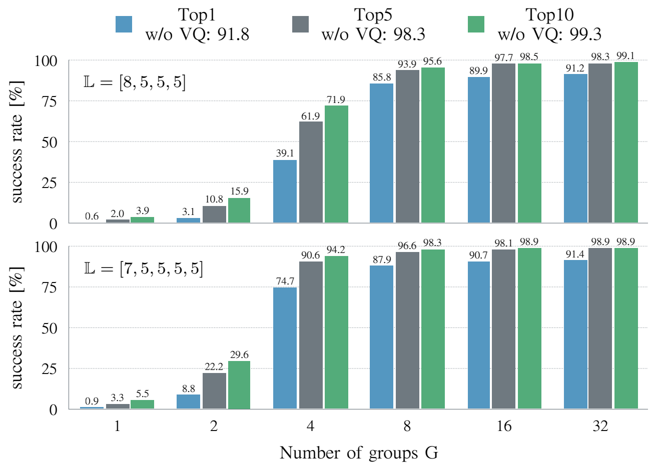


Fig. 2. Retrieval success rates for Top1, Top5, and Top10, for various FSQ settings. The baseline numbers (w/o quantization) are given in the legend.

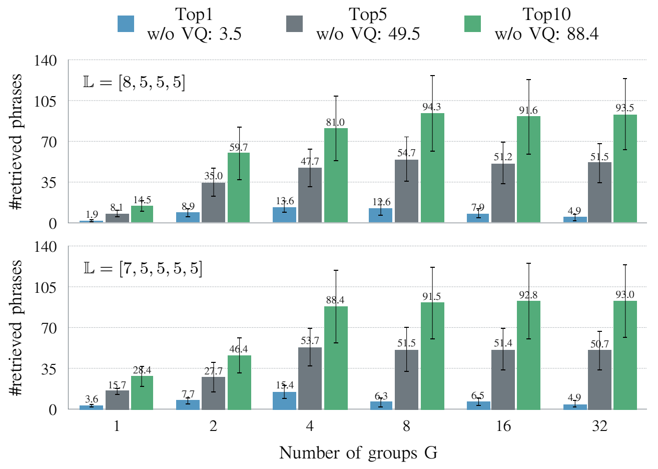


Fig. 3. Average number of phrases retrieved per utterance for Top1, Top5, and Top10 retrieval, for various FSQ settings. The baseline (w/o quantization) numbers are given in the legend. The error bars represent one standard deviation from the averages (as estimated on the 1.4k test utterances).

Success rates for the quantized models degrade a lot for $G < 8$, but appear to be very close to the non-quantized

[11]We pick two representative and recommended [34] level configurations $\mathbb{L}$ that approximately match codebook sizes equal to 2^{10} and 2^{12}, and we investigate the effect of grouping (G) in the retrieval performance. Both parameters $\mathbb{L}$ and G affect the overall capacity of our FSQ setup (see Section III).

baselines for $G \geq 16$. This shows that we can effectively use the quantization-based system to retrieve a small subset of phrases without hurting ASR accuracy (see also experiments of Section V-B). For the subsequent experiments, we mainly focus on the model with $G = 16$, $\mathbb{L} = [8, 5, 5, 5]$, as a good trade-off between retrieval accuracy and compute/memory gains based on Algorithm 1. The entire retrieved biasing list per utterance for this model contains on average 7.9 phrases (max=35) for Top1 retrieval, 51.2 phrases (max=162) for Top5 retrieval, and 91.6 phrases (max=298) for Top10 retrieval.

Since we are introducing a VQ-based approach, it is important to examine potential collisions. A collision happens when different biasing phrases get the same quantization indices, meaning that the model can no longer distinguish them. Here we estimate the collision rate as

$$\text{collision} = \frac{\#\text{unique phrases} - \#\text{unique indices}}{\#\text{unique phrases}} \tag{11}$$

for all the phrases across all the biasing catalogues in our set. Note that in any case we include exactly K phrases in the TopK retrieved list (even when multiple biasing phrases collide). As we can see in Fig. 4, the collision rate is negatively correlated with the retrieval success rate, but is consistently low for $G \geq 8$. After inspecting some collisions in those cases, most of them occur between pairs of phrases with different word ordering (e.g., *Vector Quant* vs. *Quant Vector*), different punctuation (e.g., *Vector Quant* vs. *Vector Quant.*), or slightly different spelling (e.g., *Vector* vs. *Vecctor*). The very high collision rates and, hence, the low success rates for $G = 1$ are not surprising. The subset under examination contains about 700k unique biasing phrases, whereas the maximum capacity of the quantizer for $G = 1$ is equal to $\prod_{i=1}^{|\mathbb{L}|} l_i$, which is only 1,000 for $\mathbb{L} = [8, 5, 5, 5]$ and 4,375 for $\mathbb{L} = [7, 5, 5, 5, 5]$.

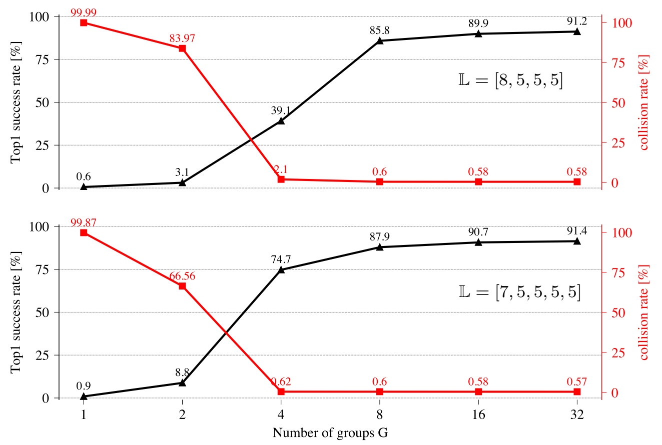


Fig. 4. Collision vs. Top1 retrieval success rate, for various FSQ settings.

B. ASR performance evaluation

We start our ASR evaluation with Table II, where the first block shows baseline results for a few NCB-enabled models depicted in Fig. 1a. In particular, we evaluate NCB consuming different numbers of biasing phrases, with $\max |\mathbb{B}| = 0$ denoting a non-biased baseline. *Dense NCB* is the system that does biasing by computing full scaled dot-product cross-attention scores. When ingesting a random selection of up to 1k or 5k biasing entries,[12] we observe an overall 21% relative WER reduction and between 50-55% relative contact NEER reduction. In this work we assume $\max |\mathbb{B}| = 5$k as a reasonable practical upper limit on the size of the biasing inventory for edge deployment, similarly to many other works using similar techniques [12], [16], [18]. However, the size of biasing catalogue may vary between users and applications in substantial ways, from a few hundred to hundreds of thousands of entries or more (e.g., when biasing to non-personal entities such as media titles), highlighting the challenge of scaling Dense NCB to uncapped biasing catalogues.

TABLE II
WER AND NEER METRICS FOR THE BASELINE AND NCB-ENABLED MODELS, AND THEIR EFFICIENT RETRIEVAL ORIENTED SETUPS.

System	$\max \lvert\mathbb{B}\rvert$	WER[%]	NEER [%]	
			Contact	Other
Dense NCB	0	7.0	39.4	15.8
Dense NCB	1k	5.6	19.7	15.4
Dense NCB	5k	5.5	17.7	15.4
(I) Retrieval NCB	5k	5.5	17.7	15.4
(II) Retrieval NCB	5k	5.5	17.7	15.4

System (I) based on $G = 16$, $\mathbb{L} = [7, 5, 5, 5, 5]$ FSQ variant.
System (II) based on $G = 16$, $\mathbb{L} = [8, 5, 5, 5]$ FSQ variant.
$\max |\mathbb{B}|$ denotes the maximum number of biasing phrases used.
Results for TopK $= 5$ (in case of Retrieval NCB), w/ CTC + attention rescoring decoder.

As shown in the previous section, quantized representation offers close-to-baseline retrieval performance for TopK ≥ 1, thus we propose to use quantization for large-scale retrieval, followed by dense biasing only for the acoustically grounded entries. This system is depicted in Fig. 1b and we will refer to this operation mode as *Retrieval NCB*. The reason we do not use, here, quantized cross-attention in a standalone, single-stage manner is that, in this case, we observed a higher confusion among the phrases with high attention scores that led to degraded accuracy, when compared to full cross-attention. We further explore this issue in Appendix A. Results for Retrieval NCB are reported in the second block of Table II, where we can observe that the retrieval-augmented systems match the performance of Dense NCB. At the same time, this approach offers much better scaling characteristics with respect to the size of the biasing catalogues, a property that we investigate in the remainder of this section.

Since the proposed technique allows us to increase the size of the biasing inventory with minimal memory footprint (see also Section V-C), Table III reports results for the case where we consume all the available contextual information. Note that from now on we use the Retrieval NCB system (II) from Table II. The first two rows repeat the accuracy results when the maximum allowed number of biasing entries is set to 1,000 and 5,000 random entries, respectively. We can see that although this change (from 1k to 5k entries) has small impact on the overall accuracy, it is important for named

[12]This means that if the size of the biasing catalogue is larger than 5k (1k), we trim it to 5k (1k) randomly selected phrases.

entity regions, as reflected by the contact NEER scores. This suggests that a biasing approach that puts constraints on the allowed size of the biasing catalogue is sub-optimal. Running decodes with all the available biasing information (maxing out at around 22.6k for the largest catalogues in the test set) does not seem to lead to further improvements, on average, over the trimmed configuration, likely due to the fact that few catalogues are affected by the 5k size threshold.

TABLE III
WER AND NEER METRICS OF RETRIEVAL NCB FOR LIMITED, FULL, AND ENUMERATED BIASING PHRASE LISTS.

Biasing Catalogue	$\max\lvert\mathbb{B}\rvert$	WER[%]	NEER [%]	
			Contact	Other
Limited	1k	5.6	19.7	15.4
Limited	5k	5.5	17.7	15.4
Full	22.6k	5.5	17.6	15.5
+ Enum. All	43.5k	5.5	16.2	15.8
+ Enum. Contacts	42.9k	5.4	15.9	15.7

Results for TopK $= 5$, w/ CTC + attention rescoring decoder.

We then experiment with enumerating the biasing phrases into word- and word-order-level combinations. For example, in a phone-book like scenario we assume that the user may utter arbitrary combinations of first and last names.[13] Such an enumeration strategy doubles the maximum inventory to around 43,000 biasing entries, but reduces NEER on contacts by 9.7% relative when compared to the non-enumerated variant. Our efficient retrieval algorithm allows us to consume fully enumerated catalogues easily, without the need to cap them due to algorithmic complexity. Applying enumerations on all the biasing entries did not bring further gains, which was expected, since for entities such as aggregated song titles, app names, etc., enumerations are less important.

Table IV shows results for different TopK retrieval settings. We observe that most of the gain is realized for $K = 1$, and contact NEER saturates after $K = 5$. The latter will be our default setting in the remainder of this work.

TABLE IV
WER AND NEER METRICS OF RETRIEVAL NCB FOR VARIOUS TOPK SETTINGS.

TopK	WER[%]	NEER [%]	
		Contact	Other
0	7.0	39.4	15.8
1	5.5	18.3	15.3
2	5.5	17.8	15.4
5	5.5	17.6	15.5
10	5.5	17.6	15.4
20	5.5	17.6	15.4

Using the *Full biasing info* system from Table III.
TopK $= 0$ essentially means that NCB is disabled.

Our proposed Retrieval NCB approach is agnostic to the exact decoding mechanism applied. While results in the previous tables were based on CTC decoding with attention rescoring, the *Retrieval NCB* rows of Table V report results with auto-regressive joint CTC-attention decoding [91].[14] We can see that Retrieval NCB works well in this configuration as well, offering an additional 16.4% relative reduction in contact NEER, or 14.8% relative reduction in general WER, when compared to the attention rescoring setup (Table III).

TABLE V
WER AND NEER METRICS FOR SYSTEMS WITH LLM FUSION AND PROMPTING AFTER JOINT CTC-ATTENTION DECODING.

System	$\max\lvert\mathbb{B}\rvert$	WER[%]	NEER [%]	
			Contact	Other
Retrieval NCB	0	6.3	38.2	12.9
+ SF NNLM	0	6.4	38.2	12.6
+ DF OpenLLaMAv2	0	6.4	37.8	11.3
++ prompt	42.9k*	5.9	31.8	11.0
Retrieval NCB	42.9k	4.6	13.3	12.5
+ SF NNLM	42.9k	4.7	13.6	12.1
+ DF OpenLLaMAv2	42.9k	4.7	13.3	11.1
++ prompt	42.9k	4.5	11.0	11.1

*used retrieved entries only via LLM prompting path for biasing.
Using the *Enum. Contacts* system from Table III.

Till now, we have considered retrieval and biasing using different configurations of the cross-attention module, as in Fig. 1a and 1b; however, shortlisted phrases can be used to contextualize the model with an arbitrary biasing machinery. One recent example relies on LLMs equipped with prompting functionality, with the biasing phrases given as a prompt to a pre-trained model [22], [23]. For our experimentation we pack the retrieved phrases into a prompt[15] and use the LLM in a delayed fusion mode [35] with the hypotheses emitted by CTC-attention decoders (Fig. 1c). Delayed fusion is a variant of shallow fusion that computes and applies LM scores for partial ASR hypotheses, but after pruning and re-tokenization of the hypotheses. This reduces the number of LLM inference calls and allows us to use an arbitrary LLM, potentially with a tokenization different than the one employed by our main ASR system. This means we do not need to re-train our ASR system to match the—typically—much larger LLM vocabulary, something that not only reduces compute cost, but also maintains the robustness of the ASR model [35], [92].

The *DF OpenLLaMAv2* rows of Table V show results with LLM delayed fusion, but without prompting. Even though there is no reduction in general WER, we can see an improvement in entity recognition. Using a robust LM especially helps with non-personal entity regions, as expected, with up to 12.4% relative NEER-other reduction. For completeness, we also report results for decodes with shallow token-level fusion with a relatively small in-domain LM, referred to as *SF NNLM*. This model is not able to be conditioned on the prompt, but helps to quantify the overall impact of LLM fusion in generic (non-prompted) mode.

The final row of the first block of the table shows the performance of the retrieval setup, when applied to shortlist biasing phrases that are packed to prompt the LLM (Fig. 1c).

[13]A single *Joe Foe* phrase becomes a set {*Joe Foe*, *Joe*, *Foe*, *Foe Joe*}.

[14]We still use the same streaming conformer acoustic encoder, but the attention module has access to the global set of audio encodings.

[15]The prompt was: *Here is a comma separated list of acoustically grounded entities you may use when predicting next relevant word: ...*

We observe that when the LLM is kept frozen and independent from the AM, the prompting performance remains somewhat limited with respect to contact NEER, with the relative improvement over baseline only being 15.8% (for comparison, Retrieval NCB got over 65% relative contact NEER reduction). Note that here we use the pre-trained weights of the LLM, without further adaptation to our use case nor allowing the LLM to access acoustic embeddings via adapter mechanism. This can perhaps explain somewhat limited gains, but LLM adaptation is out of the scope of this work.[16]

Finally, the last row of the second block of the table shows a combined variant of retrieval-based NCB where the shortlisted biasing phrases are used both as an input to a dense cross-attention module and as an LLM prompt (Fig. 1d). This gives us further 17.3% relative contact NEER reduction when compared to the strongest Retrieval NCB result, or about 71% relative when compared to the non-biased model (w/ DF OpenLLaMAv2 and $\max|\mathbb{B}| = 0$). It is interesting to observe a greater contextualization ability of LLM prompting when combined with Retrieval NCB than in a standalone LLM prompting manner (relative contact NEER reduction is 17.3% and 15.8%, respectively). This shows that for LLM fusion / re-scoring like approaches it is crucial to have high-quality biased candidate hypotheses from AM that are more amendable to LLM contextual rewrite. This also demonstrates that LLM can be leveraged effectively without acoustic encodings, and thus may potentially allow for effective conditional LLM fusion, where LLM is lazily queried for selected subsets of challenging traffic. We leave that direction for future work.

C. *Compute performance evaluation*

Fig. 5 shows the runtime evaluation of the baseline and proposed approaches for dot-product estimation. The runtime was measured with an Intel Xeon Gold 5128 processor clocked at 2.3GHz running on a single thread. In this analysis, we measured the time for computing cross-attention up to the point of (and including) the dot-product between queries and keys during inference. Any computation that can be prepared offline, such as the linear transformation of the keys in Eq. (1), is not accounted for. For the baseline approach, this includes the query preparation and the dot-product between queries and keys in Eq. (1) – (2). For the proposed approach, this includes the query preparation, the index selection, and the sum reduction as shown in Algorithm 1.

The results showcase that the proposed algorithm is faster than the baseline approach and the gains are bigger as the number of biasing phrases increases, which is expected since the time complexity is proportional to $|\mathbb{B}|$ (Section IV-B). When the biasing list has over 10k entries, all FSQ configurations achieve at least 20% reduction in runtime. For the largest biasing lists, the runtime of our proposed algorithm with the fastest FSQ setting (among the ones examined) is roughly half of the baseline. Among different FSQ configurations, smaller

[16]We also experimented with the pre-trained *Mistral-7B-v0.1* and instruction-finetuned *Mistral-7B-Instruct-v0.1* LLM variants [93]. These did not offer new insights; in particular we found both variants to have similar ability to follow prompt contextualization instructions on our task.

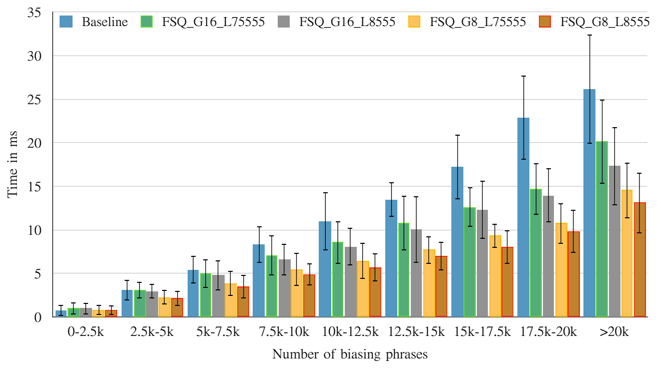


Fig. 5. Runtime analysis of baseline and proposed approaches across various biasing list sizes for all the utterances in our test set. The error bars represent one standard deviation from the averages, computed from all the queries in the same bin.

number of groups, G, and fewer levels, $|\mathbb{L}|$, are contributing factors. This is consistent with the complexity analysis in Table I, where we saw that runtime complexity is proportional to both those parameters. Also note that for all the models in Fig. 5, $|\mathbb{L}| \cdot G < D = 256$.

Fig. 6 shows the memory usage analysis of the baseline and proposed approaches. For this analysis, we picked a query with about one second of audio and roughly 10k biasing phrases. Then, we artificially created larger biasing lists by repetition. The purpose of doing so is to compare memory usage in extreme hypothetical scenarios with huge biasing lists. Similar to the runtime analysis, we measured the memory usage for computing cross-attention up to the point of dot-product between queries and keys. In this analysis, we used full precision for floating point numbers. For indices, since we only need 3 bits for each level (to represent up to $\max l_i = 8$ values), we can use a 16-bit integer to represent all the elements e_i^g in $\boldsymbol{e}^g$, for every group $g \in \{1, 2, \ldots, G\}$ (Eq. (4)). Given that this particular FSQ configuration has 16 groups, each biasing phrase only needs 32 bytes, which is substantially smaller than the space needed for the baseline approach.

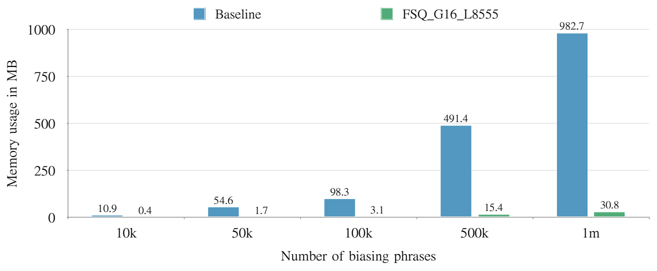


Fig. 6. Memory usage of baseline and proposed approaches across various biasing list sizes. The FSQ configuration has $G = 16$ groups and $\mathbb{L} = [8, 5, 5, 5]$ levels.

The results show that our proposed algorithm uses little memory compared to the baseline, as expected. Consider the case with 1 million biasing phrases: we need $16 \times 2 \times 1$ million bytes, i.e., 30.5MB of memory, to represent all the biasing phrases. In comparison, the baseline approach would need $256 \times 4 \times 1$ million bytes, i.e. 976.6MB of memory to represent the whole biasing list. Even if we reduced the precision for the biasing phrases from 32-bit to 8-bit, the memory needed for the baseline approach would still be much larger than our quantization-based algorithm (244MB vs. 31MB). Therefore, depending on the precision, our approach achieves over 85-95% reduction in memory usage when the biasing list has over 10k entries. This memory reduction is due to a much more compact representation of the biasing phrases, as explained in Section IV-B. However, we would like to mention that to achieve the full memory reduction, one would need to implement a kernel that combines index selection, sum reduction and TopK retrieval as in Algorithm 1. Without this kernel, the memory reduction is smaller; we discuss this in detail in Appendix B.

VI. Conclusions

We proposed an efficient approximation to cross-attention that uses vector quantization techniques, allowing us to ground large biasing catalogues quickly on audio data with limited memory footprint. We incorporated this mechanism in a retrieval setup, where the shortlisted phrases are used to bias an ASR system through dense cross-attention and/or through LLM prompting. We evaluated our proposed methods on a large set, with our strongest system yielding a relative contact NEER reduction of about 71% and a relative WER reduction of about 30%, when compared to a non-biased baseline.

Our quantization-based technique allows the ASR model to leverage more biasing entries that would otherwise be discarded due to excessive compute and memory cost. We demonstrated the need for employing larger biasing catalogues by reporting sub-optimal results when we limited the initial catalogues to a few thousand phrases, a common practice in the literature. Our retrieval approach opens up the way to scale NCB up to a much larger number of biasing phrases, something essential in several scenarios; for example, when the goal is to bias the ASR model on the media domain (including all the aggregated song and album titles).

While we analyzed and evaluated our system using FSQ, the ideas we described can potentially be paired with alternative efficient vector quantization algorithms and retrieval-oriented losses (e.g., [94]). Additionally, we believe that further improvements can be obtained by relatively simple enhancements. For instance, for all the LLM-based results reported in this work we used a publicly available pre-trained LLM. The overall accuracy could be improved after fine-tuning such an LLM in the speech domain and adapting it for our use case. Moreover, even though we followed the common practice of employing a separate transformer-based context encoder, we could exploit the representational capabilities of the LLM even more, by processing the biasing phrases through it in an LLM2Vec-like manner [95]. This could potentially further reduce the overall memory footprint, by discarding all the additional context encoder parameters.

Acknowledgments

We would like to thank Stefan Braun, Erik McDermott, Xinwei Li and Stephen Pulman for their feedback on this work. Thanks to Xiao Zhang for providing an in-house NNLM.

Appendix A
Single-Stage Quantized NCB

Throughout this paper we have explored two-stage NCB systems where we used our quantization-based approach to estimate dot-products between queries (acoustic embeddings) and keys (biasing embeddings) in order to retrieve the TopK biasing phrases with the highest scores. In this appendix we study a single-stage NCB approach where the biasing module of Dense NCB (Fig. 1a) is fully replaced by its quantized counterpart; we refer to this NCB mode as *Quantized NCB*. The new quantized module approximates Eq. (2) using discretized contextual encodings, introduced in Section III, for both keys and values. Representing the keys, $\boldsymbol{K}$, and the values, $\boldsymbol{V}$, by their quantized approximations, $\tilde{\boldsymbol{V}}$ and $\tilde{\boldsymbol{K}}$, respectively, the biasing encodings can be simply estimated as

$$\tilde{\boldsymbol{Y}} = \mathrm{softmax}\left(\alpha\, \boldsymbol{Q}\tilde{\boldsymbol{K}}^{\top}\right)\tilde{\boldsymbol{V}} \tag{12}$$

Those biasing encodings, $\tilde{\boldsymbol{Y}}$, are added to the acoustic embeddings and fed into the ASR decoders. The dot-products in $\boldsymbol{Q}\tilde{\boldsymbol{K}}^{\top}$ are still estimated using the algorithm described in Section IV, but without the need for TopK retrieval.

Table VI repeats the results of Table II for Dense NCB (Fig. 1a) and Retrieval NCB (Fig. 1b) systems and compares their performance to their Quantized NCB equivalents. Even though Quantized NCB performs better that the non-biased baseline (with $\max|\mathbb{B}| = 0$), we can see that the quantization-based approximation, when used in a standalone manner, offers substantially worse biasing accuracy, compared to either dense cross-attention or retrieval-oriented two-stage setups.

TABLE VI
WER and NEER metrics for the baseline and NCB-enabled models, their quantized approximations, and their retrieval oriented setups.

System	$\max\|\mathbb{B}\|$	WER[%]	NEER [%]	
			Contact	Other
Dense NCB	0	7.0	39.4	15.8
Dense NCB	5k	5.5	17.7	15.4
(I) Quantized NCB	5k	6.2	26.6	16.0
(II) Quantized NCB	5k	6.3	28.6	15.9
(I) Retrieval NCB	5k	5.5	17.7	15.4
(II) Retrieval NCB	5k	5.5	17.7	15.4

Systems (I) based on $G = 16$, $\mathbb{L} = [7, 5, 5, 5, 5]$ FSQ variant.
Systems (II) based on $G = 16$, $\mathbb{L} = [8, 5, 5, 5]$ FSQ variant.
Results for TopK = 5 (in case of Retrieval NCB), w/ CTC + attention rescoring decoder.

We observed that, on average, the attention scores of the quantized systems have lower values and a less peaky behavior compared to dense cross-attention. In other words, the attention probability mass, after the softmax function of Eq. (2) or Eq. (12), is distributed across more biasing phrases in the case of quantized approaches, which can lead to higher confusion among different phrases. This also explains the stats provided in Fig. 3, where we can see that the number of Top1 phrases retrieved per utterance, on average, is much higher for the quantized systems. For instance, for the setup we have mostly studied through our experiments in Section V, with $G = 16$, $\mathbb{L} = [8, 5, 5, 5]$, we have 7.9 phrases retrieved on average per utterance, compared to only 3.5 phrases in the case of non-quantized cross-attention. Note that this behavior can be especially problematic during frames without spoken entities where the model is expected to attend to the back-off token. This is because if the model does not attend as much as it should to the back-off, it can yield over-biased transcriptions. However, as long as the right phrases are included in the retrieved set (i.e., the success rate—or recall—is high enough), we can successfully use the quantization-based, retrieval-oriented systems without hurting ASR accuracy.

Appendix B
Implementation and Memory Consumption

In this appendix, we explain why a direct implementation of Algorithm 1 in PyTorch could not achieve full memory reduction. This is due to the fact that PyTorch API does not allow performing index selection, sum reduction and retrieval at the same time. See a simplified code snippet of Algorithm 1 implemented in PyTorch,

```
# K is the number of TopK entries for retrieval
# T is the length of acoustic encoder frame sequence
# B is the number of biasing phrases
# G is the number of groups
# L is the depth of the FSQ level
# U is the number of different possible values in
    all FSQ levels
# e contains the indices with shape [B, G*L]
# S_qan is the score tensor with shape [T, G*L*U]
# result is the output tensor with shape [T, K]

e = e.view(-1)
a = torch.index_select(S_qan, dim=-1, index=e)
a = a.view(T, B, G*L)
s = torch.sum(a, dim=-1)
s = s.view(T, B)
result = torch.topk(s, K, dim=1)
```

The problem is that the index selection step would create an intermediate tensor, `a`, of shape `[T, B*L*G]`, which is dominated by `B` for large biasing catalogues. For instance, for a typical setting in our experiments we could have `B=10000`, `L=4`, `G=16`, and `T=33` for a 2sec audio (200 frames with 6-fold downsampling). Additionally, the TopK operation is performed only after we have the entire score tensor, `s`. That said, the memory consumption in that case is still smaller than the baseline approach that needs $O(|\mathbb{B}|D)$ space, since $D > G \cdot |\mathbb{L}|$ in a typical setting. To achieve full memory reduction, however, one would need to implement a kernel that performs index selection, sum reduction and TopK retrieval at the same time as in Algorithm 1. Fig. 7 compares the memory usage with or without this custom kernel, using the same assumptions as in Section V-C.

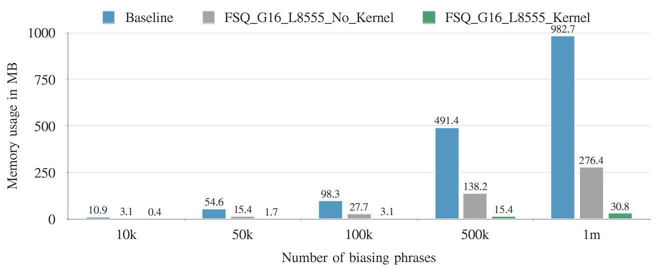


Fig. 7. Memory usage of baseline and our proposed approach with or without a custom kernel to perform index selection, sum reduction and retrieval at the same time. The FSQ uses $G = 16$ groups and $\mathbb{L} = [8, 5, 5, 5]$ levels.

References

[1] R. Prabhavalkar, T. Hori, T. N. Sainath *et al.*, "End-to-end speech recognition: A survey," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 325–351, 2024.

[2] J. Guo, T. N. Sainath, and R. J. Weiss, "A spelling correction model for end-to-end speech recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 5651–5655.

[3] T. N. Sainath, R. Prabhavalkar, S. Kumar *et al.*, "No need for a lexicon? Evaluating the value of the pronunciation lexica in end-to-end models," in *Proc. ICASSP*. IEEE, 2018, pp. 5859–5863.

[4] U. Alon, G. Pundak, and T. N. Sainath, "Contextual speech recognition with difficult negative training examples," in *Proc. ICASSP*. IEEE, 2019, pp. 6440–6444.

[5] D. Zhao, T. N. Sainath, D. Rybach *et al.*, "Shallow-fusion end-to-end contextual biasing," in *Proc. Interspeech*. ISCA, 2019, pp. 1418–1422.

[6] D. Le, G. Keren, J. Chan *et al.*, "Deep shallow fusion for RNN-T personalization," in *Proc. SLT*. IEEE, 2021, pp. 251–257.

[7] K. B. Hall, E. Cho, C. Allauzen *et al.*, "Composition-based on-the-fly rescoring for salient n-gram biasing," in *Proc. Interspeech*. ISCA, 2015, pp. 1418–1422.

[8] I. Williams, A. Kannan, P. S. Aleksic *et al.*, "Contextual speech recognition in end-to-end neural network systems using beam search," in *Proc. Interspeech*. ISCA, 2018, pp. 2227–2231.

[9] E. McDermott, H. Sak, and E. Variani, "A density ratio approach to language model fusion in end-to-end automatic speech recognition," in *Proc. ASRU*. IEEE, 2021, pp. 434–441.

[10] Z. Meng, N. Kanda, Y. Gaur *et al.*, "Internal language model estimation for domain-adaptive end-to-end speech recognition," in *Proc. ICASSP*. IEEE, 2021, pp. 7338–7342.

[11] G. Pundak, T. N. Sainath, R. Prabhavalkar *et al.*, "Deep context: End-to-end contextual speech recognition," in *Proc. SLT*. IEEE, 2018, pp. 418–425.

[12] F. Chang, J. Liu, M. Radfar *et al.*, "Context-aware transformer transducer for speech recognition," in *Proc. ASRU*. IEEE, 2021, pp. 503–510.

[13] K. M. Sathyendra, T. Muniyappa, F. Chang *et al.*, "Contextual adapters for personalized speech recognition in neural transducers," in *Proc. ICASSP*. IEEE, 2022, pp. 8537–8541.

[14] A. Bruguier, R. Prabhavalkar, G. Pundak, and T. N. Sainath, "Phoebe: Pronunciation-aware contextualization for end-to-end speech recognition," in *Proc. ICASSP*. IEEE, 2019, pp. 6171–6175.

[15] M. Jain, G. Keren, J. Mahadeokar *et al.*, "Contextual RNN-T for open domain ASR," in *Proc. Interspeech*. ISCA, 2020, pp. 11–15.

[16] G. Sun, C. Zhang, and P. C. Woodland, "Tree-constrained pointer generator for end-to-end contextual speech recognition," in *Proc. ASRU*. IEEE, 2021, pp. 780–787.

[17] T. Munkhdalai, K. C. Sim, A. Chandorkar *et al.*, "Fast contextual adaptation with neural associative memory for on-device personalized speech recognition," in *Proc. ICASSP*. IEEE, 2022, pp. 6632–6636.

[18] Z. Meng, Z. Wu, R. Prabhavalkar *et al.*, "Text injection for neural contextual biasing," in *Proc. Interspeech*. ISCA, 2024, pp. 2985–2989.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, pp. 1735–1780, 1997.

[20] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *Proc. NeuRIPS*, 2017, pp. 5998–6008.

[21] S. Minaee, T. Mikolov, N. Nikzad *et al.*, "Large language models: A survey," *arXiv preprint arXiv:2402.06196*, 2024.

[22] X. Gong, A. Lv, Z. Wang, and Y. Qian, "Contextual biasing speech recognition in speech-enhanced large language model," in *Proc. Interspeech*. ISCA, 2024, pp. 257–261.

[23] M. Wang, W. Han, I. Shafran *et al.*, "SLM: Bridge the thin gap between speech and text foundation models," in *Proc. ASRU*. IEEE, 2023.

[24] Z. Chen, H. Huang, A. Andrusenko *et al.*, "SALM: Speech-augmented language model with in-context learning for speech recognition and translation," in *Proc. ICASSP*. IEEE, 2024, pp. 13 521–13 525.

[25] Z. Lei, X. Na, M. Xu *et al.*, "Contextualization of ASR with LLM using phonetic retrieval-based augmentation," in *Proc. ICASSP*. IEEE, 2025.

[26] A. Alexandridis, K. M. Sathyendra, G. Strimel *et al.*, "Gated contextual adapters for selective contextual biasing in neural transducers," in *Proc. ICASSP*. IEEE, 2023.

[27] Z. Yang, S. Sun, X. Wang *et al.*, “Two stage contextual word filtering for context bias in unified streaming and non-streaming transducer,” in *Proc. Interspeech*. ISCA, 2023, pp. 3257–3261.

[28] M. Levy, A. Jacoby, and Y. Goldberg, “Same task, more tokens: The impact of input length on the reasoning performance of large language models,” in *Proc. ACL (Volume 1: Long Papers)*. ACL, 2024, pp. 15 339–15 353.

[29] S. Tong, P. Harding, and S. Wiesler, “Slot-triggered contextual biasing for personalized speech recognition using neural transducers,” in *Proc. ICASSP*. IEEE, 2023.

[30] Z. Chen, M. Jain, Y. Wang *et al.*, “Joint grapheme and phoneme embeddings for contextual end-to-end ASR.” in *Proc. Interspeech*. ISCA, 2019, pp. 3490–3494.

[31] D. Kulshreshtha, S. Dingliwal, B. Houston, and S. Bodapati, “Multilingual contextual adapters to improve custom word recognition in low-resource languages,” in *Proc. Interspeech*. ISCA, 2023, pp. 3302–3306.

[32] Y. Sudo, M. Shakeel, Y. Fukumoto *et al.*, “Contextualized automatic speech recognition with attention-based bias phrase boosted beam search,” in *Proc. ICASSP*. IEEE, 2024, pp. 10 896–10 900.

[33] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” in *Proc. NeurIPS*, vol. 30, 2017, p. 6309–6318.

[34] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen, “Finite scalar quantization: VQ-VAE made simple,” in *Proc. ICLR*, 2024.

[35] T. Hori, M. Kocour, A. Haider *et al.*, “Delayed fusion: Integrating large language models into first-pass decoding in end-to-end speech recognition,” in *Proc. ICASSP*. IEEE, 2025.

[36] B. Yusuf, A. Gourav, A. Gandhe, and I. Bulyko, “On-the-fly text retrieval for end-to-end ASR adaptation,” in *Proc. ICASSP*. IEEE, 2023.

[37] M. Wang, I. Shafran, H. Soltau *et al.*, “Speech-to-text adapter and speech-to-entity retriever augmented LLMs for speech understanding,” *arXiv preprint arXiv:2306.07944*, 2023.

[38] X. Gong, A. Lv, Z. Wang *et al.*, “BR-ASR: Efficient and scalable bias retrieval framework for contextual biasing ASR in speech LLM,” *arXiv preprint arXiv:2505.19179*, 2025.

[39] Z. Wu, G. Song, C. Li *et al.*, “Deferred NAM: Low-latency top-k context injection via deferred context encoding for non-streaming ASR.” in *Proc. NAACL*. ACL, 2024, pp. 315–323.

[40] Z. Huang, D. Caseiro, K. Joshi *et al.*, “Optimizing large-scale context retrieval for end-to-end ASR,” in *Proc. Interspeech*. ISCA, 2024, pp. 4573–4577.

[41] S. Dingliwal, M. Sunkara, S. Ronanki *et al.*, “Personalization of CTC speech recognition models,” in *Proc. SLT*. IEEE, 2023, pp. 302–309.

[42] Y. Peng, Y. Sudo, M. Shakeel, and S. Watanabe, “OWSM-CTC: An open encoder-only speech foundation model for speech recognition, translation, and language identification,” in *Proc. ACL (Volume 1: Long Papers)*. ACL, 2024, pp. 10 192–10 209.

[43] R. Huang, M. Yarmohammadi, S. Khudanpur, and D. Povey, “Improving neural biasing for contextual speech recognition by early context injection and text perturbation,” in *Proc. Interspeech*. ISCA, 2024, pp. 752–756.

[44] S. D. Torres, A. Sen, A. Rana *et al.*, “Promptformer: Prompted conformer transducer for ASR,” in *Proc. ICASSP*. IEEE, 2024, pp. 11 821–11 825.

[45] K. Huang, A. Zhang, Z. Yang *et al.*, “Contextualized end-to-end speech recognition with contextual phrase prediction network,” in *Proc. Interspeech*. ISCA, 2023, pp. 4933–4937.

[46] S. M. Jayanthi, D. Kulshreshtha, S. Dingliwal *et al.*, “Retrieve and copy: Scaling ASR personalization to large catalogs,” in *Proc. EMNLP: Industry Track*. ACL, 2023, pp. 631–639.

[47] S. Zhou, Z. Li, Y. Hong *et al.*, “CopyNE: Better contextual ASR by copying named entities,” in *Proc. ACL (Volume 1: Long Papers)*. ACL, 2024, pp. 2675–2686.

[48] M. A. Jalal, P. P. Parada, G. Pavlidis *et al.*, “Locality enhanced dynamic biasing and sampling strategies for contextual ASR,” in *Proc. ASRU*. IEEE, 2023.

[49] M. Bleeker, P. Swietojanski, S. Braun, and X. Zhuang, “Approximate nearest neighbour phrase mining for contextual speech recognition,” in *Proc. Interspeech*. ISCA, 2023, pp. 939–943.

[50] T. N. Sainath, R. Prabhavalkar, D. Caseiro *et al.*, “Improving contextual biasing with text injection,” in *Proc. ICASSP*. IEEE, 2023.

[51] S. Kim, T. Hori, and S. Watanabe, “Joint CTC-attention based end-to-end speech recognition using multi-task learning.” in *Proc. ICASSP*. IEEE, 2017, pp. 4835–4839.

[52] P. K. Rubenstein, C. Asawaroengchai, D. D. Nguyen *et al.*, “AudioPaLM: A large language model that can speak and listen,” *arXiv preprint arXiv:2306.12925*, 2023.

[53] F. Seide, M. Doulaty, Y. Shi *et al.*, “Speech ReaLLM–real-time streaming speech recognition with multimodal LLMs by teaching the flow of time,” in *Proc. Interspeech*. ISCA, 2024, pp. 1900–1904.

[54] X. Yang, W. Kang, Z. Yao *et al.*, “PromptASR for contextualized ASR with controllable style,” in *Proc. ICASSP*. IEEE, 2024, pp. 10 536–10 540.

[55] E. Lakomkin, C. Wu, Y. Fathullah *et al.*, “End-to-end speech recognition contextualization with large language models,” in *Proc. ICASSP*. IEEE, 2024, pp. 12 406–12 410.

[56] C. Chen, R. Li, Y. Hu *et al.*, “It’s never too late: Fusing acoustic information into large language models for automatic speech recognition,” in *Proc. ICLR*, 2024.

[57] K. Hu, T. N. Sainath, B. Li *et al.*, “Massively multilingual shallow fusion with large language models,” in *Proc. ICASSP*. IEEE, 2023.

[58] L. Xu, Y. Gu, J. Kolehmainen *et al.*, “RescoreBERT: Discriminative speech recognition rescoring with BERT,” in *Proc. ICASSP*. IEEE, 2022, pp. 6117–6121.

[59] C.-H. H. Yang, Y. Gu, Y.-C. Liu *et al.*, “Generative speech recognition error correction with large language models and task-activating prompting,” in *Proc. ASRU*. IEEE, 2023.

[60] A. Radford, J. W. Kim, T. Xu *et al.*, “Robust speech recognition via large-scale weak supervision,” in *Proc. ICML*. PMLR, 2023, pp. 28 492–28 518.

[61] A. Robatian, M. Hajipour, M. R. Peyghan *et al.*, “GEC-RAG: Improving generative error correction via retrieval-augmented generation for automatic speech recognition systems,” *arXiv preprint arXiv:2501.10734*, 2025.

[62] Z. Niu, G. Zhong, and H. Yu, “A review on the attention mechanism of deep learning,” *Neurocomputing*, vol. 452, pp. 48–62, 2021.

[63] T. Dao, D. Fu, S. Ermon *et al.*, “FlashAttention: Fast and memory-efficient exact attention with IO-awareness,” *Proc. NeuRIPS*, vol. 35, pp. 16 344–16 359, 2022.

[64] H. Liu, M. Zaharia, and P. Abbeel, “Ring attention with blockwise transformers for near-infinite context,” in *NeurIPS Foundation Models for Decision Making Workshop*, 2023.

[65] D. Le, M. Jain, G. Keren *et al.*, “Contextualized streaming end-to-end speech recognition with trie-based deep biasing and shallow fusion,” in *Proc. Interspeech*. ISCA, 2021, pp. 1772–1776.

[66] M. Han, L. Dong, Z. Liang *et al.*, “Improving end-to-end contextual speech recognition with fine-grained contextual knowledge selection,” in *Proc. ICASSP*. IEEE, 2022, pp. 8532–8536.

[67] T. Munkhdalai, Z. Wu, G. Pundak *et al.*, “NAM+: Towards scalable end-to-end contextual biasing for adaptive ASR,” in *Proc. SLT*. IEEE, 2023, pp. 190–196.

[68] Z. Wu, T. Munkhdalai, P. Rondon *et al.*, “Dual-mode NAM: Effective Top-K context injection for end-to-end ASR,” in *Proc. Interspeech*. ISCA, 2023, pp. 221–225.

[69] J. Fu, J. Liu, H. Tian *et al.*, “Dual attention network for scene segmentation,” in *Proc. CVPR*. IEEE/CVF, 2019, pp. 3141–3149.

[70] S. Y. Sahai, J. Liu, T. Muniyappa *et al.*, “Dual-attention neural transducers for efficient wake word spotting in speech recognition,” in *Proc. ICASSP*. IEEE, 2023.

[71] A. Gourav, J. Kolehmainen, P. Shivakumar *et al.*, “Multi-modal retrieval for large language model based speech recognition,” in *Findings ACL*. ACL, 2024, pp. 4435–4446.

[72] R. Gray, “Vector quantization,” *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 4–29, 1984.

[73] A. Razavi, A. van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with VQ-VAE-2,” in *Proc. NeurIPS*, vol. 32, 2019.

[74] N. Zeghidour, A. Luebs, A. Omran *et al.*, “Soundstream: An end-to-end neural audio codec,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 495–507, 2022.

[75] D. Yang, S. Liu, R. Huang *et al.*, “HiFi-codec: Group-residual vector quantization for high fidelity audio codec,” *arXiv preprint arXiv:2305.02765*, 2023.

[76] J. Peng, D. Liu, S. Xu, and H. Li, “Generating diverse structure for image inpainting with hierarchical VQ-VAE,” in *Proc. CVPR*. IEEE/CVF, 2021, pp. 10 775–10 784.

[77] D. Lee, C. Kim, S. Kim *et al.*, “Autoregressive image generation using residual quantization,” in *Proc. CVPR*. IEEE/CVF, 2022, pp. 11 523–11 532.

[78] A. Roy, A. Vaswani, A. Neelakantan, and N. Parmar, “Theory and experiments on vector quantized autoencoders,” *arXiv preprint arXiv:1805.11063*, 2018.

[79] A. Łańcucki, J. Chorowski, G. Sanchez *et al.*, “Robust training of vector quantized bottleneck models,” in *Proc. IJCNN*. IEEE, 2020.

[80] J. Yu, X. Li, J. Y. Koh *et al.*, “Vector-quantized image modeling with improved VQGAN,” in *Proc. ICLR*, 2022.

[81] L. Yu, J. Lezama, N. B. Gundavarapu *et al.*, “Language model beats diffusion - tokenizer is key to visual generation,” in *Proc. ICLR*, 2024.

[82] J. Zhou, S. Zhao, Y. Liu *et al.*, “KNN-CTC: Enhancing ASR via retrieval of CTC pseudo labels,” in *Proc. ICASSP*. IEEE, 2024, pp. 11 006–11 010.

[83] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv preprint arXiv:1308.3432*, 2013.

[84] P. Swietojanski, S. Braun, D. Can *et al.*, “Variable attention masking for configurable transformer transducer speech recognition,” in *Proc. ICASSP*. IEEE, 2023.

[85] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. ICLR*, 2014.

[86] D. S. Park, W. Chan, Y. Zhang *et al.*, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in *Proc. Interspeech*. ISCA, 2019, pp. 2613–2617.

[87] T. Kudo and J. Richardson, “SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” in *Proc. EMNLP: System Demonstrations*. ACL, 2018, pp. 66–71.

[88] X. Geng and H. Liu, “OpenLLaMA: An open reproduction of LLaMA,” May 2023. [Online]. Available: https://github.com/openlm-research/open_llama

[89] H. Touvron, T. Lavril, G. Izacard *et al.*, “LLaMA: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

[90] Z. Yao, D. Wu, X. Wang *et al.*, “WeNet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit,” in *Proc. Interspeech*. IEEE, 2021, pp. 4054–4058.

[91] T. Hori, S. Watanabe, and J. Hershey, “Joint CTC/attention decoding for end-to-end speech recognition,” in *Proc. ACL (Volume 1: Long Papers)*. ACL, 2017, pp. 518–529.

[92] Y. Higuchi, B. Yan, S. Arora *et al.*, “BERT meets CTC: New formulation of end-to-end speech recognition with pre-trained masked language model,” in *Findings EMNLP*. ACL, 2022, pp. 5486–5503.

[93] A. Q. Jiang, A. Sablayrolles, A. Mensch *et al.*, “Mistral 7B,” *arXiv preprint arXiv:2310.06825*, 2023.

[94] R. Guo, P. Sun, E. Lindgren *et al.*, “Accelerating large-scale inference with anisotropic vector quantization,” in *Proc. ICML*. PMLR, 2020, pp. 3887–3896.

[95] P. BehnamGhader, V. Adlakha, M. Mosbach *et al.*, “LLM2Vec: Large language models are secretly powerful text encoders,” *Proc. COLM*, 2024.