DUPLEXKV: UNDERSTANDING AND LEARNING KV CACHE EVICTION IN FULL-DUPLEX SPEECH MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Full-duplex spoken dialogue models listen and speak at once, and every audio frame of both speakers appends key-value states to the cache, so its memory grows with elapsed session time, even during silence, rather than only with the amount of speech exchanged. We first analyze what this cache holds on two full-duplex models, MiniCPM-o 4.5 and Raon-SpeechChat. The cost of losing a token depends not only on its recency and accumulated attention but also on its content and on whether the token belongs to the user or to the agent. Eviction under a budget can also leave the model taking the turn while the reply loses its content, a failure that the next-token loss tracks poorly because the loss scores a given reply token by token rather than the reply that the model generates under eviction. We propose DuplexKV, a lightweight L_0-Gate trained jointly with the backbone under a fixed cache budget with an L_0-regularized objective and combined with attention and recency at inference, together with Judged Eviction Tuning (JET), which tunes the gate on an LLM judge's verdicts about the replies the model generates. On the user interruption task of Full-Duplex-Bench, DuplexKV with JET attains the highest judge score among prior eviction policies on both backbones even at small budgets, responding to almost every interruption, and its advantage grows as conversations become longer.

1 INTRODUCTION

Full-duplex spoken dialogue models listen and speak at the same time, so a user can interrupt, backchannel or take the turn while the model is still talking (Défossez et al., 2024; Veluri et al., 2024; Cui et al., 2026; Kim et al., 2026). The audio of both speakers enters the model at a fixed frame rate, and every frame appends tokens to the key-value cache whether or not anyone is speaking. The cache of a session therefore grows with elapsed time, even during silence, rather than only with the amount of speech exchanged. Across sessions, the memory a server needs grows with the number of sessions it holds at once (Zhang et al., 2023). Long conversations and many concurrent users are possible only with a bounded cache. Commercial realtime speech services already bound the context, and they do so by discarding the oldest content first. Audio sessions are either limited to 15 minutes unless the oldest tokens are removed by a sliding window (Google, 2026), or continued while the oldest messages are truncated once the context window is full (OpenAI, 2025).

Discarding the oldest tokens first assumes that they are the least needed. The KV cache eviction literature replaces this assumption with finer criteria. Training-free methods rank cached tokens by statistics derived from attention (Zhang et al., 2023; Li et al., 2024; Oren et al., 2024). Learned methods train the ranking toward a target retention ratio, an attention-derived importance target, or the output of the Full KV cache (Łańcucki et al., 2025; Bui et al., 2026b; Huang et al., 2024; Zhou et al., 2026). These criteria were developed mainly for text language models. Their extensions to images (Chen et al., 2024; Song et al., 2026) and audio (Wang et al., 2026c; Jia et al., 2026) compress an input that precedes the response, not a stream to which two speakers write at the same time. The cache of a full-duplex model instead interleaves the audio of two speakers, much of it written during silence, with a few tokens that govern turn-taking. We therefore ask what a full-duplex cache holds and whether recency, attention or a learned target reflects what the model needs to keep.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

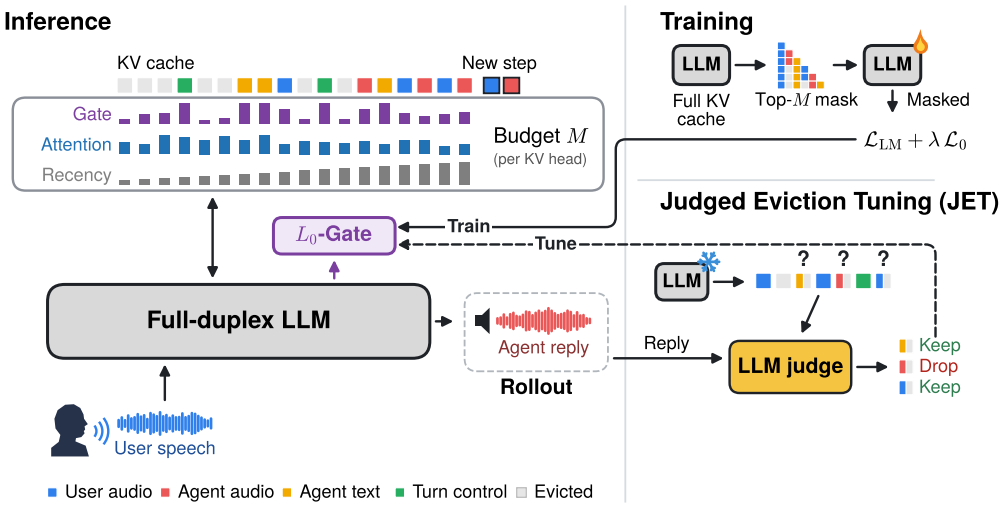

Figure 1: Overview of DuplexKV and JET. At inference, each KV head keeps the M tokens with the highest rank-aggregated score of the gate, attention and recency. In JET the LLM judge labels only the tokens that some KV heads retain and others evict, marked with a question mark.

We examine this question on two full-duplex models with opposite speech-generation designs, MiniCPM-o 4.5 and Raon-SpeechChat. MiniCPM-o 4.5 decodes the speech of the agent from hidden states outside the cache, whereas Raon-SpeechChat writes the speech of the agent back into the cache, and both models write the user and the agent to separate positions of the cache. We find that the cost of losing a token depends on its content and on whether the token belongs to the user or to the agent, and that turn-control tokens, a few percent of the cache, are among the most costly to lose. Existing policies rank tokens by the time of writing or by accumulated attention, which leave out the content and the speaker of a token. Eviction under a budget can also leave the model taking the turn while the reply loses its content, a failure that the next-token loss tracks poorly because the loss scores a given reply token by token rather than the reply that the model generates under eviction.

Motivated by these findings, we propose DuplexKV (Figure 1), which learns what a full-duplex model should keep in its KV cache under the budget of inference. A lightweight L_0-Gate is trained jointly with the backbone under a fixed cache budget, so that its loss reflects the evictions the gate itself causes. Because this loss scores a given reply rather than the reply generated under eviction, we further introduce Judged Eviction Tuning (JET), which tunes the gate on an LLM judge's verdicts about the reply the model actually produced. Language models already decide which memories an agent retains (Park et al., 2023; Packer et al., 2023), and preference labels produced by language models train dialogue models about as effectively as human labels (Lee et al., 2024), which suggests that the verdicts of an LLM judge are a usable training signal.

We evaluate DuplexKV on the user interruption task of Full-Duplex-Bench v1.0 (FDB v1.0) against prior eviction policies. With JET, DuplexKV attains the highest judge score on both backbones even at small budgets, responding to almost every interruption, whereas on Raon-SpeechChat competitive baselines leave many interruptions unanswered. On MiniCPM-o 4.5 sessions of up to 24 minutes, its advantage over the baselines grows as the session lengthens.

Our contributions are as follows.

- We analyze what the KV cache of a full-duplex model contains and what losing each part of it costs, on two backbones with opposite speech-generation designs.
- We propose DuplexKV, an L_0-Gate trained jointly with the backbone under a cache budget and combined with attention and recency, whose gain grows in long sessions.
- We introduce JET, which tunes the gate on LLM judge verdicts at the eviction boundary, and DuplexKV + JET outperforms prior policies on both backbones.

2 PRELIMINARIES

2.1 FULL-DUPLEX GENERATION

We adopt the formalism of Chen & Yu (2025). An agent $\mathcal{A}$ interacts with an environment $\mathcal{E}$, and a discretizer $\mathcal{T}$ maps the continuous audio of both speakers onto two sequences aligned on a common frame index $t = 1, \ldots, T$,

$$S^{\mathcal{E}} = (e_1, \ldots, e_T), \qquad S^{\mathcal{A}} = (a_1, \ldots, a_T). \tag{1}$$

In the conditional formulation, the agent emits

$$a_t \sim P\big(a_t \mid S^{\mathcal{E}}_{\leq t},\, S^{\mathcal{A}}_{<t};\, \theta\big). \tag{2}$$

We write $\mathcal{X}_t = S^{\mathcal{E}}_{\leq t} \cup S^{\mathcal{A}}_{<t}$ for the session tokens on which the agent conditions at step t, and call one increment of the common frame index t a step. Frames are produced regardless of whether the user or the agent is speaking, and T grows with the duration of the session rather than with the number of turns.

2.2 CACHED TOKENS AND TYPES

We write j for a cached token, that is, the key and value vectors written at one position of the sequence, $\tau(j) \in \{1, \ldots, T\}$ for the step in which the token was written, and $\rho(j)$ for the type of the token. Let $\mathcal{K}_t$ denote the set of tokens retained after step t, and let M denote the budget in tokens per KV head.

The five types are user audio, agent audio, agent text, turn control and the prefix $\mathcal{P}$ (Appendix B). Turn control consists of the special tokens that govern turn-taking, and the prefix holds the system prompt cached before the session begins, does not count toward M and is never evicted. We name the audio types by modality rather than by speech, because a token is written at every frame, including frames in which the corresponding speaker is silent.

2.3 FULL-DUPLEX BACKBONES

We study MiniCPM-o 4.5 (Cui et al., 2026) and Raon-SpeechChat (Kim et al., 2026), two open-weight models that write the streams of the user and the agent to separate positions of a single cache, which an analysis by token type requires (Appendix B). Moshi (Défossez et al., 2024), by contrast, sums the embeddings of all streams into one position per frame, so the audio of one speaker cannot be retained without the other. The two backbones differ in where the speech of the agent is generated, which sets the types that occupy the cache and their rates. MiniCPM-o 4.5 writes ten user audio tokens, two chunk markers, one turn-control token, at most one stop token and the agent text of each one-second step, and decodes the speech of the agent from hidden states outside the cache. Raon-SpeechChat writes one user audio token, one agent audio token and at most one agent text or turn-control token, the turn state, in each 80 ms frame, so the audio of the agent enters the cache at every frame, including frames in which the agent listens. A session of n seconds therefore writes at least $13n$ tokens on MiniCPM-o 4.5 and $25n$ on Raon-SpeechChat (Table 3). Without eviction the cache grows linearly with elapsed time, and since the length of a streaming session is unknown while the session runs, the budget is an absolute number of tokens rather than a fraction of the prompt length as in long-context inference.

3 ANALYSIS OF THE FULL-DUPLEX KV CACHE

We measure what the KV cache of a full-duplex model contains and what losing each part costs on MiniCPM-o 4.5 and Raon-SpeechChat, and derive from the measurements the design of DuplexKV and JET, using the judge score and the takeover rate (TOR) defined in Section 5.1.

3.1 RECENCY AND THE SLIDING WINDOW

At small budgets the most recent tokens do not hold everything a reply needs. Sliding-window services and StreamingLLM (Xiao et al., 2024) assume that the tokens a reply needs are recent. A window of the M most recent tokens covers M divided by the token rate, 2.0 s of the session on

MiniCPM-o 4.5 and 1.2 s on Raon-SpeechChat at $M = 32$ (Appendix I.1). Inference with recency as the only channel on the backbones trained with DuplexKV (Section 4) lowers the judge score below that of DuplexKV by 2.13 at $M = 32$ and 0.37 at $M = 64$ on MiniCPM-o 4.5 and by 2.39 and 0.80 on Raon-SpeechChat (Table 7). We therefore keep recency as a channel, and the rest of this section examines the older tokens.

3.2 Cost of removing each token type

The cost of losing a token depends on its content and its stream rather than on its share of the cache. In KV cache eviction for LLMs, structural tokens are evicted first by a learned retention score (Bui et al., 2026b) and retained longest by attention-based ranking (Oren et al., 2024). We remove all tokens of one type from the Full KV cache and compare the judge score and the takeover rate on FDB v1.0 user interruption, where audio fills 64.8% of the cache on MiniCPM-o 4.5 and 90.4% on Raon-SpeechChat (Figure 2).

On MiniCPM-o 4.5, removing the turn-control tokens, 6.5% of the session, lowers the judge score by 4.03, the same drop as removing all user audio, and on Raon-SpeechChat removing the turn state, 4.0% of the session, lowers the takeover rate from 0.935 to 0.050. These results favor retaining turn-control tokens over evicting them first. Among audio tokens the stream matters as well as speech activity. On Raon-SpeechChat, removing the user audio written while the user is silent lowers the judge score by 1.86, removing the agent audio written while the agent listens, 25.4% of the session, lowers it by 0.05, and removing the agent audio written while the agent speaks lowers it by 3.44.

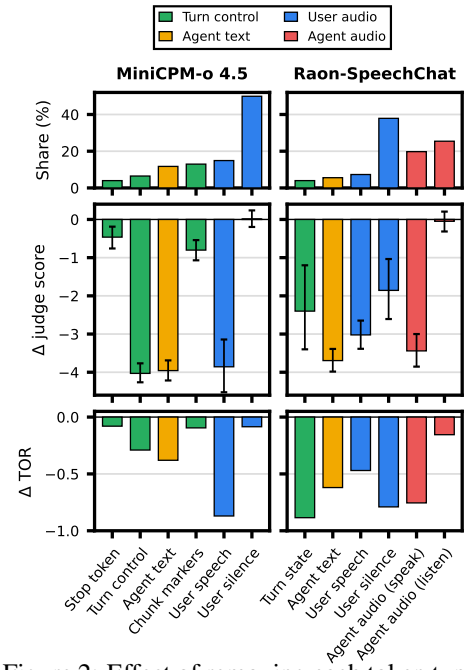

Figure 2: Effect of removing each token type from the Full KV cache on FDB v1.0 user interruption, with 95% paired intervals.

Ranking by accumulated attention (Zhang et al., 2023) assumes that this statistic measures importance, but in a growing cache the number of queries that a token receives grows with the number n_j of tokens written after it. Under the Full KV cache, the accumulated attention increases with n_j for every token type on both backbones, whereas the mean attention per query decreases for most types (Appendix I.3), so accumulated attention favors old tokens, the opposite of recency. Selecting among the older tokens requires the content and the stream of each token, which motivates a score computed from the representation of each token.

3.3 Failure modes under a budget

Learned eviction methods are trained on the next-token loss or on agreement with the output of the Full KV cache (Łańcucki et al., 2025; Bui et al., 2026b; Zhou et al., 2026; Althoubi, 2026), and the language-modeling term of our objective is of the same kind, so these objectives presume that the damage of an eviction appears in the likelihood of the reply. Removing a whole type of token lowers the takeover rate and the judge score together, whereas eviction under a budget can leave turn-taking intact while the reply loses its content, becoming fragmentary or answering a different question (Figure 16 in Appendix I.5). Across our settings, replies that keep the takeover rate within 0.05 of the Full KV cache but lose at least one point of judge score arise only under budget eviction, on both backbones, and in none of the 16 type removals. When we force the reply generated with the Full KV cache under each intervention, the change in loss on its word tokens has a rank correlation with the change in judge score of -0.76 on MiniCPM-o 4.5 and -0.23 on Raon-SpeechChat (Appendix I.4), and the loss tracks the score least where turn-taking stays intact. On Raon-SpeechChat, for example, inference without the recency channel raises the loss by 2.23 at $M = 32$ but by only 0.11 at $M = 256$, where the judge score still falls by 2.59 with the takeover rate within 0.02. On MiniCPM-o 4.5, removing the chunk markers and stop tokens lowers the judge

score by 1.36 while the loss on word tokens falls by 0.10. The loss scores the likelihood of a fixed reply, whereas these failures lie in the reply the model generates, so JET takes its training signal from that reply (Section 4.5).

4 DUPLEXKV

4.1 PROBLEM FORMULATION

We seek a retained set $\mathcal{K}_t \subseteq \mathcal{X}_t$ with $|\mathcal{K}_t| \le M$ for which $P(a_t \mid \mathcal{K}_t; \theta)$ approximates $P(a_t \mid \mathcal{X}_t; \theta)$. Writing $z_j^{(t)} \in \{0, 1\}$ for the indicator that token j is retained after step t, the corresponding training problem is

$$\min_{z} \ \mathcal{L}_{\mathrm{LM}}(z) \quad \text{subject to} \quad \textstyle\sum_{j:\,\tau(j)\le t} z_j^{(t)} \le M \ \text{ and } \ z_j^{(t)} \le z_j^{(t-1)} \quad \text{for all } t. \tag{3}$$

Here $\mathcal{L}_{\mathrm{LM}}(z)$ is the next-token loss on the output tokens of the agent when each step attends only to the retained tokens, defined in Eq. (6). The second constraint prevents an evicted token from returning to the cache, since a policy that may reselect any M past tokens at each step must store all of them. This problem cannot be solved directly, for two reasons. First, z is not a parameter that can be learned, because each token is created at inference time within a session that was not observed during training. Second, the constraint is combinatorial. We address the first by amortizing z into an L_0-Gate that scores each token from its representation (Section 4.2), and the second by imposing the constraint exactly during training with a top-M rule on the gate score, with gradients reaching the gate through a straight-through estimator (Section 4.3). At inference, the gate is combined with attention and recency in each KV head (Section 4.4). Because $\mathcal{L}_{\mathrm{LM}}$ scores the recorded agent turn rather than the generated reply, we introduce JET, which tunes the trained gate on an LLM judge's verdicts about tokens at the eviction boundary (Section 4.5).

4.2 AMORTIZED L_0-GATE

Let $\mathbf{h}_j \in \mathbb{R}^d$ be the last-layer representation of token j, computed with the Full KV cache during training and under eviction at inference. A single head $g_{\boldsymbol{\phi}}$, shared across layers and KV heads, maps the representation of each token to a scalar retention logit,

$$\log \alpha_j = g_{\boldsymbol{\phi}}(\mathbf{h}_j). \tag{4}$$

Because $g_{\boldsymbol{\phi}}$ reads the representation rather than storing a value per token, it assigns a logit to tokens of sessions that were never seen during training. The head consists of an RMSNorm and a single linear projection, which adds fewer than 10^4 parameters to the backbone.

4.3 TRAINING UNDER A CACHE BUDGET

Budget mask. During training, the cache retains the M tokens with the largest gate logits among those still present and evicts the others. Writing $\mathcal{N}_t$ for the tokens written at step t, the retained set evolves as

$$\mathcal{K}_t = \mathrm{Top}_M\left(\mathcal{K}_{t-1} \cup \mathcal{N}_t;\, \log \alpha\right), \qquad \mathcal{K}_0 = \emptyset, \tag{5}$$

where $\mathrm{Top}_M(\mathcal{S}; \log \alpha)$ returns the M elements of $\mathcal{S}$ with the largest $\log \alpha$, or all of $\mathcal{S}$ when $|\mathcal{S}| \le M$. The indicator $z_j^{(t)} = \mathbf{1}[j \in \mathcal{K}_t]$ therefore satisfies the constraint of Eq. (3) at every step by construction.

A query at step t attends to the prefix, to $\mathcal{K}_t$ and to the tokens of steps $t-1$ and t, which are exempt from the budget mask, and all other tokens are masked out of attention. The mask is shared by all layers and KV heads. The rule in Eq. (5) has no useful gradient, so we use the straight-through estimator. The forward pass uses the hard indicator $z_j^{(t)}$, and the backward pass replaces its derivative with that of $\sigma(\log \alpha_j)$ (Appendix C). The gate is thus trained on the loss induced by its own ranking. We build the training mask from the gate logit rather than from the combined score of Eq. (10), so that every eviction during training is a decision of the gate (Appendix C). Under the combined score most training evictions would be decided by attention and recency, and the straight-through estimator would still pass their cost to the gate logits, fitting the gate to correct the two statistics rather than to rank tokens by the cost of losing them. Figure 7 in Appendix C shows the resulting mask on both backbones.

Two-pass training without rollout. An ordinary forward pass retains every token and therefore provides no signal about the cost of an eviction. The mask cannot be derived within the pass that it applies to, because a parallel forward needs the mask of every step before its first layer is computed, whereas the gate reads the last layer. Deriving the mask step by step would require one forward per step, as in a rollout. We instead run the backbone twice, each time over the whole session at once. The first pass keeps the Full KV cache, carries no gradient (sg, stop-gradient), and supplies the representations $\mathbf{h}_j$ from which the gate scores every token. The second pass runs the backbone under the mask that those scores imply, and the losses below are taken on its output. For a fixed mask, the second pass matches step-by-step decoding with the same evictions under teacher forcing, since every query attends to the same keys at the same positions, except that an evicted key keeps an attention weight scaled by $\epsilon = 10^{-6}$ (Eq. (16)) and the tokens of the preceding step stay visible (Appendix I.6). Beyond these two exceptions, the approximation lies in the scores. During training the gate reads representations computed with the Full KV cache, whereas at inference the gate reads representations computed under eviction, and the two yield the same mask whenever they place the same M tokens above the cut. Algorithm 1 in Appendix C summarizes one training step.

L_0-regularized objective. The objective combines two terms. The first is the next-token loss of the masked pass on the agent's output tokens,

$$\mathcal{L}_{\mathrm{LM}} = -\frac{1}{Z} \sum_{i \in \mathcal{O}} \omega_{\rho(i)} \log P\big(x_i \mid x_{\mathcal{R}_i};\, \theta\big), \tag{6}$$

where $\mathcal{O}$ is the set of agent output tokens, $\mathcal{R}_i$ is the set of earlier positions visible to token i under the budget mask, $\omega_{\rho(i)}$ is the loss weight of its type, and Z normalizes over the batch as in the released training code of each backbone. On Raon-SpeechChat the next-token loss also covers the audio codes of the agent (Appendix G). This term is the only one that measures the value of the retained tokens, and reaches the gate through the straight-through estimator.

For the second term we follow Louizos et al. (2018) and treat each token as having a gate that is open with probability

$$p_j = \sigma\left(\log \alpha_j - \beta \log \frac{-\gamma}{\zeta}\right), \tag{7}$$

which is the probability that a hard concrete variable with location $\log \alpha_j$, temperature β and stretch interval (γ, ζ) is nonzero. Averaging over the set $\mathcal{V}$ of tokens that the gate scores gives

$$\mathcal{L}_0 = \frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} p_j, \tag{8}$$

the expected fraction of open gates. Because the budget is already enforced by Eq. (5), this term does not set how many tokens are retained. The L_0 term penalizes retention and lowers the logits of tokens that the language-modeling loss does not support, which separates the scores of tokens below the budget cut. The gate and the backbone are trained jointly on

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda\, \mathcal{L}_0. \tag{9}$$

4.4 Rank-aggregated eviction at inference

Let $\ell = 1, \ldots, L$ index layers and $h = 1, \ldots, H$ index KV heads. The gate is one of three channels, and each KV head ranks its candidate tokens by a weighted sum of their percentile ranks,

$$s_j^{(\ell,h)} = w_g\, \Pi^{(\ell,h)}(\log \alpha)_j + w_a\, \Pi^{(\ell,h)}(A^{(\ell,h)})_j + w_d\, \Pi^{(\ell,h)}(d)_j. \tag{10}$$

Here $A_j^{(\ell,h)}$ is the attention that token j has accumulated over the queries of steps $\tau(j)$ to t, summed over the query heads grouped with KV head h of layer ℓ, and $d_j = -n_j$, where n_j is the number of tokens written after token j, so that $\Pi^{(\ell,h)}(d)$ ranks the most recent tokens highest. The operator $\Pi^{(\ell,h)}$ maps a channel to its percentile rank among the candidates of that head, with tied values sharing their mean rank, so each weight sets the share of the decision given to its channel rather than a scale on the channel's own units. The tokens $\mathcal{N}_t$ of the current step, which are not yet candidates, are kept and count toward the budget, and each head fills the remaining slots with the earlier tokens of the largest score,

$$\mathcal{K}_t^{(\ell,h)} = \mathcal{N}_t \cup \mathrm{Top}_{M-|\mathcal{N}_t|}\left(\mathcal{K}_{t-1}^{(\ell,h)};\, s^{(\ell,h)}\right), \tag{11}$$

and the remaining tokens are evicted from that head's cache. We refer to Eqs. (10) and (11) as the eviction policy of DuplexKV. The gate and recency channels take the same raw value in every layer and head, so differences between the retained sets of heads originate in the attention channel.

4.5 JUDGED EVICTION TUNING

Of the three channels in Eq. (10), attention and recency are statistics of the cache that record where attention has been placed and how long a token has been present. Attention is also known to concentrate on tokens that carry little content, because the softmax must assign its full probability mass even when no key is relevant (Xiao et al., 2024). Whether a token is retained therefore need not reflect whether it is needed. The gate is the only channel trained to score tokens, and therefore the only channel through which the policy can be trained toward a need that neither statistic measures.

The objective of Eq. (9) does not supply that signal, because its target is the recorded agent turn rather than the reply that the model generates after an eviction. Section 3.3 shows that this loss tracks the failures of the generated reply poorly. We therefore obtain the importance of a token from an LLM judge that assesses its effect on the generated reply.

Judged tokens. We roll out the trained gate under the eviction policy at budget M and read the retained sets at the step $t^\star$ at which the reply begins. Writing $z_j^{(\ell,h)} = \mathbf{1}\big[\, j \in \mathcal{K}_{t^\star}^{(\ell,h)}\big]$, the share of layers and heads that retain token j is

$$\bar{z}_j = \frac{1}{LH} \sum_{\ell=1}^{L} \sum_{h=1}^{H} z_j^{(\ell,h)}, \tag{12}$$

and the tokens submitted to the judge are

$$\mathcal{J} = \{\, j \in \mathcal{U}_r \;:\; 0 < \bar{z}_j < 1 \,\}, \tag{13}$$

with $\mathcal{U}_r$ the user audio and agent text tokens written in the judged turn r, the two types a judge can read as words. We take these tokens as the eviction boundary of the policy, where the KV heads differ in retaining the token. Turn-control tokens are excluded because these tokens carry the turn-taking structure rather than the content of the dialogue, which is what the judge assesses. For $j \in \mathcal{J}$, the scores $s_j^{(\ell,h)}$ fall on both sides of the retention threshold across heads, and since $\log \alpha_j$ enters the score of every head, a perturbation of $\log \alpha_j$ can alter the retention of j at inference. For a token with $\bar{z}_j \in \{0, 1\}$ all heads agree, and a verdict on it carries less information about the ranking.

Verdicts. Before the judged turn r, no token is evicted, and the recorded dialogue fills the Full KV cache. The eviction policy runs only during turn r and stops at $t^\star$, so any change in the reply is caused by evictions within turn r, and every verdict refers to a token that exists in every rollout. Tokens are grouped into spoken words by their time alignment, and each word w becomes one item. The judge receives the dialogue, the generated reply and the list of items, and judges for each item whether the reply shows that the item was needed. The verdict is binary, $y_w = 1$ when the item should have been kept, and every token inherits the verdict of its word, $y_j = y_{w(j)}$ (Appendix F.3). The verdicts are collected once with the trained gate before tuning and stay fixed while the gate is tuned.

JET objective. The gate is tuned on the verdicts with a class-balanced cross-entropy on $\sigma(\log \alpha_j)$ over the set $\mathcal{B}$ of judged tokens in a batch,

$$\mathcal{L}_{\mathrm{JET}} = -\frac{1}{|\mathcal{B}|} \sum_{j \in \mathcal{B}} \omega_{y_j} \Big[\, y_j \log \sigma(\log \alpha_j) + (1 - y_j) \log \big(1 - \sigma(\log \alpha_j)\big) \Big], \tag{14}$$

where $\omega_c = |\mathcal{B}| / (2\,|\{j \in \mathcal{B} : y_j = c\}|)$ gives the two verdicts equal total weight in each batch. The gate is initialized from the gate trained with Eq. (9), the backbone is frozen, and each step is the training step of Algorithm 1 with the objective

$$\mathcal{L} = \mathcal{L}_{\mathrm{LM}} + \lambda\, \mathcal{L}_0 + \eta\, \mathcal{L}_{\mathrm{JET}}. \tag{15}$$

Algorithm 2 and Figure 11 in Appendix F give the procedure.

Table 1: Comparison with eviction policies on FDB v1.0 user interruption at $M \in \{32, 64\}$, with the judge score averaged over the episodes in which the model takes the turn.

		MiniCPM-o 4.5				Raon-SpeechChat			
		$M = 32$		$M = 64$		$M = 32$		$M = 64$	
	Method	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)
	Full KV	4.03	0.96	4.03	0.96	4.01	0.94	4.01	0.94
Rec.	StreamingLLM (Xiao et al., 2024)	3.19	<u>0.99</u>	3.71	<u>0.99</u>	1.17	0.33	1.61	0.48
Attn.	H2O (Zhang et al., 2023)	0.43	0.95	3.84	0.96	0.88	0.89	2.10	<u>0.98</u>
	TOVA (Oren et al., 2024)	3.36	0.97	4.25	0.95	0.45†	0.06	0.41†	0.09
	SnapKV (Li et al., 2024)	3.74	<u>0.99</u>	4.08	0.97	2.00	0.56	4.10	0.83
Alloc.	Ada-KV (Feng et al., 2025)	3.77	0.96	4.20	0.98	2.27	0.47	4.13	0.80
Key	R-KV (Cai et al., 2025a)	4.07	0.97	4.29	0.96	<u>3.78</u>	0.59	3.88	0.94
	KeyDiff (Park et al., 2026)	1.22	0.88	2.52	0.96	0.44†	0.04	1.44	0.42
Audio	AudioKV (Wang et al., 2026c)	4.29	0.96	4.31	0.98	2.01	0.42	<u>4.12</u>	0.80
	VoxZip (Jia et al., 2026)	0.47	0.80	3.59	<u>0.99</u>	0.57	**0.98**	1.88	0.90
Learned	DMS (Łańcucki et al., 2025)	3.16	<u>0.99</u>	3.95	<u>0.99</u>	0.87	0.30	3.38	0.80
	TrimKV (Bui et al., 2026b)	4.06	0.98	4.35	0.99	3.18	0.51	<u>4.12</u>	0.62
	DuplexKV (Ours)	<u>4.37</u>	**1.00**	<u>4.39</u>	**1.00**	3.53	<u>0.97</u>	4.07	<u>0.98</u>
	DuplexKV + JET (Ours)	**4.46**	**1.00**	**4.44**	**1.00**	**3.90**	**0.98**	**4.27**	**0.99**

Bold and underline mark the highest and second highest value in each column. † Fewer than 30 episodes with TOR $= 1$.

5 EXPERIMENTS

5.1 SETUP

Backbones and training. We apply DuplexKV to MiniCPM-o 4.5 and Raon-SpeechChat and train the gate jointly with the language model of each backbone on Behavior-SD at $M = 64$ (Appendix G). Because the next-token loss tracks the reply failures of Section 3.3 poorly, we select λ and the inference weights by the judge score on interruption episodes built from the validation split of Behavior-SD (Table 5 in Appendix G).

Evaluation. We evaluate on the user interruption task of FDB v1.0, whose 200 episodes each end with the user interrupting the model. For the ablation we also evaluate on 185 interruption episodes built from the validation split of Behavior-SD (Appendix D.2). The takeover rate (TOR) is the fraction of episodes in which the model takes the turn, and the judge score is the mean rating of the replies in those episodes, given by an LLM judge with the released prompt (Appendix E, with the agreement with a second judge in Appendix E.1). Budgets range over $M \in \{32, 64, 128, 256\}$ tokens per KV head, and every token written after the prefix, including chunk markers and turn-control tokens, counts toward M. Differences between two methods are paired over the episodes in which both take the turn, with 95% bootstrap intervals.

Baselines. We compare with the eviction policies of Table 1, grouped by the quantity they rank, and list every baseline in Table 6. All policies keep the prefix and decide at every step of the stream under the budget accounting of DuplexKV (Appendix H). Training-free policies and the Full KV cache run on the released weights of each backbone, learned policies are trained on Behavior-SD (Lee et al., 2025), and DuplexKV runs on the backbone trained jointly with its gate.

5.2 COMPARISON WITH EVICTION POLICIES ACROSS BUDGETS

On MiniCPM-o 4.5, DuplexKV attains the highest judge score among the baselines at $M = 32$, 64 and 256 and ties the highest at $M = 128$, while taking the turn in every episode, and JET raises it further at $M = 32$ and $M = 64$. On Raon-SpeechChat, where audio fills 90.4% of the cache, the baselines that keep a high judge score at $M = 32$ do so on a subset of the episodes, as R-KV takes the turn in 59% of the episodes. DuplexKV + JET takes the turn in 98% and attains the highest judge score at both budgets of Table 1.

Setting $w_g = 0$ in Eq. (10) while keeping w_a and w_d on the same jointly trained backbone lowers the judge score of DuplexKV in all eight combinations of backbone, evaluation set and $M \in \{32, 64\}$ (Appendix I.7). The decrease is largest on Raon-SpeechChat at $M = 32$, 0.80 on Behavior-SD and 0.94 on FDB v1.0, so a ranking by attention and recency does not reproduce the ranking of DuplexKV at small budgets. Run on the jointly trained backbone of DuplexKV, four baselines of

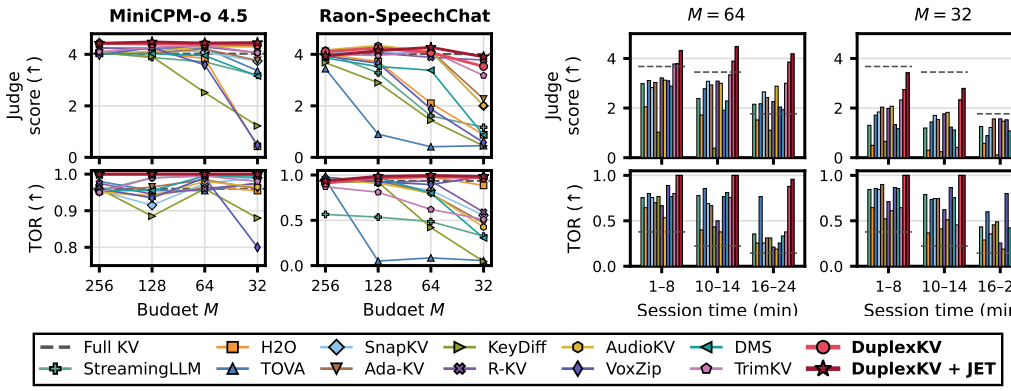

Figure 3: Comparison with eviction policies across budgets on FDB v1.0 user interruption.

Figure 4: Comparison with eviction policies over otoSpeech sessions with MiniCPM-o 4.5.

Table 1, StreamingLLM, SnapKV, R-KV and AudioKV, stay below DuplexKV + JET at $M = 32$ and 64 on both backbones (Table 7 in Appendix H).

5.3 Tuning the gate on judge verdicts

We collect verdicts from Qwen3.8-27B on 1,000 Behavior-SD training dialogues per backbone rolled out at $M = 32$, and train the gate on them with the backbone frozen (Appendix F). JET raises the judge score at the budgets where eviction decides the most, from 3.53 to 3.90 at $M = 32$ and from 4.07 to 4.27 at $M = 64$ on Raon-SpeechChat, and from 4.37 to 4.46 at $M = 32$ on MiniCPM-o 4.5 (Appendix F.4).

On Raon-SpeechChat, JET lowers the share of KV heads that retain the agent's own earlier content words at the reply step from 7.8% to 3.5% at $M = 32$, and the replies whose score JET raises contain a word of the request more often, 14% against 6%, and reuse a content word of the earlier topic less often, 19% against 47% (Figure 14 in Appendix F.5).

5.4 Eviction over long sessions

The episodes of FDB v1.0 last under a minute, so we evaluate on the 58 task-oriented dialogues of otoSpeech (oto.earth, 2026), whose sessions last up to 24 minutes. Raon-SpeechChat takes almost no turn in these sessions from 4 minutes on even with the Full KV cache (Appendix D.2), so we compare eviction policies on MiniCPM-o 4.5. At each of nine session times from 1 to 24 minutes, the model receives 30 recorded dialogues up to a turn boundary and generates the next 30 s, and the reply is scored as in FDB v1.0 (Figures 4 and 9).

At $M = 64$ the judge score of DuplexKV stays between 3.79 and 3.89 across session time, while the highest baseline falls from 3.78 to 3.00, so the gap grows from 0.01 to 0.85 as the recorded history lengthens. At $M = 32$ DuplexKV takes every turn, whereas the baselines take 19 to 80% of the turns after 16 minutes. With JET, the judge score at $M = 64$ rises to between 4.19 and 4.48 across the three ranges of session time, above the Full KV cache with the released weights in each.

6 Conclusion

A full-duplex model writes to its cache even in silence, so a bounded cache is what allows long sessions and many concurrent users. Our analysis shows that the tokens worth keeping in this cache are set by their content and stream, and that a wrong eviction shows more in the reply the model generates than in the likelihood of a given reply. DuplexKV learns what to keep under the inference budget, JET brings a judgment of the generated reply into training, and extending this judged signal to the timing of backchannels and turn transitions is left to future work.

ETHICS STATEMENT

The training data, Behavior-SD, and the evaluation episodes of FDB v1.0 are spoken dialogues synthesized with text-to-speech, so no recordings of human participants are used. The method reduces the memory that a full-duplex model needs to serve long sessions and many concurrent users, and it introduces no capability beyond that of the backbone it is applied to. Eviction can change what a model remembers of a conversation, and a deployment that relies on the retained context should be evaluated on its own data.

REPRODUCIBILITY STATEMENT

Appendix D describes the training data and the construction of both evaluation sets, and Appendix E gives the evaluation protocol, including the judge prompt, the judge model and the statistics. Appendix G lists every training and inference setting, Algorithm 1 and Algorithm 3 give the training step and the inference procedure, and Appendix I.2 derives the memory figures from the model configurations. The supplementary material contains the implementation of the gate, the budget mask, the training step, the eviction policy and the construction of JET labels, together with the settings of both backbones. The code will be released upon publication.

7 USE OF LARGE LANGUAGE MODELS

We use an LLM judge to rate the replies in every evaluation (Appendix E) and to produce the verdicts on which JET trains the gate (Section 4.5). We also use LLMs to assist with code, experiments and writing.

8 LICENSES

MiniCPM-o 4.5 is released under the Apache 2.0 license, and Raon-SpeechChat and Full-Duplex-Bench under CC BY-NC 4.0. Behavior-SD, otoSpeech and Parakeet-TDT-0.6B-v2, the speech recognizer of Full-Duplex-Bench, are released under CC BY 4.0. Qwen3.8-27B, the JET judge, is released under Apache 2.0 and Silero VAD under MIT. We use them for non-commercial research in accordance with these licenses.

REFERENCES

Jinwoo Ahn, Ingyu Seong, Akhil Kedia, Junhan Kim, Hyemi Jang, Kangwook Lee, and Yongkweon Jeon. LookaheadKV: Fast and accurate KV cache eviction by glimpsing into the future without generation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RVLMGPXt2i.

Asaad Althoubi. DistillCache: Kl-guided adaptive KV-cache eviction for memory-efficient LLM inference. *arXiv preprint arXiv:2608.08878*, 2026.

Ngoc Bui, Hieu Trung Nguyen, Arman Cohan, and Rex Ying. Make each token count: Towards improving long-context performance with KV cache eviction. *arXiv preprint arXiv:2605.09649*, 2026a.

Ngoc Bui, Shubham Sharma, Simran Lamba, Saumitra Mishra, and Rex Ying. Cache what lasts: Token retention for memory-bounded KV cache in LLMs. In *International Conference on Learning Representations*, volume 2026, pp. 120316–120342, 2026b.

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-KV: Redundancy-aware KV cache compression for reasoning models. In *Advances in Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=2jwAjomEDB.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=ayi7qezU87.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024.

Qian Chen, Yafeng Chen, Yanni Chen, Mengzhe Chen, Yingda Chen, Chong Deng, Zhihao Du, Ruize Gao, Changfeng Gao, Zhifu Gao, et al. MinMo: A multimodal large language model for seamless voice interaction. *arXiv preprint arXiv:2501.06282*, 2025.

Yuxuan Chen and Haoyuan Yu. From turn-taking to synchronous dialogue: A survey of full-duplex spoken language models. *arXiv preprint arXiv:2509.14515*, 2025.

Junbo Cui, Bokai Xu, Chongyi Wang, Tianyu Yu, Weiyue Sun, Yingjing Xu, Tianran Wang, Zhihui He, Wenshuo Ma, Tianchi Cai, et al. MiniCPM-o 4.5: Towards real-time full-duplex omni-modal interaction. *arXiv preprint arXiv:2604.27393*, 2026.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: KV cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025.

Zican Dong, Peiyu Liu, Junyi Li, Zhipeng Chen, Han Peng, Shuo Wang, and Xin Zhao. ForesightKV: Optimizing KV cache eviction for reasoning models by learning long-term contribution. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=znV8JHv8b8.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=tcisuhGsQZ.

Yuan Feng, Haoyu Guo, Junlin Lv, S Kevin Zhou, and Xike Xie. DefensiveKV: Taming the fragility of KV cache eviction in LLM inference. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=nJgS06sX3O.

Yuan Feng, Junlin Lv, Haoyu Guo, Yukun Cao, S Kevin Zhou, and Xike Xie. CriticalKV: Optimizing KV cache eviction from an output perturbation perspective. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=vMwlcLyI4b.

Anastasiia Filippova, David Grangier, Marco Cuturi, and Joao Monteiro. Stochastic KV routing: Enabling adaptive depth-wise cache sharing. *arXiv preprint arXiv:2604.22782*, 2026.

Chaoyou Fu, Haojia Lin, Xiong Wang, YiFan Zhang, Yunhang Shen, Xiaoyu Liu, Haoyu Cao, Zuwei Long, Heting Gao, Ke Li, Long Ma, Xiawu Zheng, Rongrong Ji, Xing Sun, Caifeng Shan, and Ran He. VITA-1.5: Towards GPT-4o level real-time vision and speech interaction. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8PUzLga3lU.

Gabriel Garcia. Protection is (nearly) all you need: Structural protection dominates scoring in globally capped KV eviction. *arXiv preprint arXiv:2605.18053*, 2026.

Google. Live API best practices, Gemini API. https://ai.google.dev/gemini-api/docs/live-api/best-practices, 2026. Accessed 2026-09-23.

Lukas Hauzenberger, Niklas Schmidinger, Anamaria-Roberta Hartl, David Stap, Thomas Schmied, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. KVpop–Key-Value cache compression with predictive online pruning. *arXiv preprint arXiv:2607.05061*, 2026.

Yuxiang Huang, Binhang Yuan, Xu Han, Chaojun Xiao, and Zhiyuan Liu. Locret: Enhancing eviction in long-context LLM inference with trained retaining heads on consumer-grade devices. *arXiv preprint arXiv:2410.01805*, 2024.

Wenxu Jia, Dongjie Fu, Xize Cheng, Fangming Feng, Linjun Li, Wenshi Chen, Yingming Li, Zhou Zhao, and Tao Jin. VoxZip: Semantic-anchored temporal KV cache compression for long-context audio inference. *arXiv preprint arXiv:2608.08569*, 2026.

Freeman Jiang, Ramon Sanabria, Soham Deshmukh, Bandhav Veluri, Simon Michael Vuch Williams, Elliott K Suen, Garreth Lee, Kevin Yoonho Choi, Takuya Umeki, Riku Kubo, et al. TurnBench: A multi-domain benchmark for turn-taking dynamics in spoken dialogue. *arXiv preprint arXiv:2608.25218*, 2026.

Beomsoo Kim, Changho Choi, Dohyun Kim, Dongki Lee, Ethan Ewer, Eunchong Kim, Gyeongman Kim, Haechan Kim, Hyeonghwan Kim, Inkyu Park, et al. Raon-Speech technical report. *arXiv preprint arXiv:2605.23912*, 2026.

Adrian Łańcucki, Konrad Staniszewski, Piotr Nawrot, and Edoardo Ponti. Inference-time hyper-scaling with KV cache compression. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=8ZiElzQxf1.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. RLAIF vs. RLHF: scaling reinforcement learning from human feedback with AI feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

Sehun Lee, Kang-wook Kim, and Gunhee Kim. Behavior-SD: Behaviorally aware spoken dialogue generation with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9574–9593, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970, 2024.

Guan-Ting Lin, Jiachen Lian, Tingle Li, Qirui Wang, Gopala Anumanchipalli, Alexander H Liu, and Hung-yi Lee. Full-Duplex-Bench: A benchmark to evaluate full-duplex spoken dialogue models on turn-taking capabilities. In *2025 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8. IEEE, 2025.

Guan-Ting Lin, Chen Chen, Zhehuai Chen, and Hung-yi Lee. Full-Duplex-Bench-v3: Benchmarking tool use for full-duplex voice agents under real-world disfluency. *arXiv preprint arXiv:2604.04847*, 2026a.

Guan-Ting Lin, Shih-Yun Shan Kuan, Jiatong Shi, Kai-Wei Chang, Siddhant Arora, Shinji Watanabe, and Hung-yi Lee. Full-Duplex-Bench-v2: A multi-turn evaluation framework for duplex dialogue systems with an automated examiner. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 27–36, 2026b.

Guan-Ting Lin, Shih-Yun Shan Kuan, Qirui Wang, Jiachen Lian, Tingle Li, Shinji Watanabe, and Hung-yi Lee. Full-Duplex-Bench v1. 5: Evaluating overlap handling for full-duplex speech models. In *ICASSP 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 19447–19451. IEEE, 2026c.

Alexander H Liu, Andy Ehrenberg, Andy Lo, Chen-Yo Sun, Guillaume Lample, Jean-Malo Delignon, Khyathi Raghavi Chandu, Patrick von Platen, Pavankumar Reddy Muddireddy, Rohin Arora, et al. Voxtral realtime. *arXiv preprint arXiv:2602.11298*, 2026.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Liuyue, Bo Li, Xuming Hu, and Xiaowen Chu. ChunkKV: Semantic-preserving KV cache compression for efficient long-context LLM inference. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=20JDhbJqn3.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32332–32344. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bz.html.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo Ponti. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37396–37412. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/nawrot24a.html.

Tu Anh Nguyen, Eugene Kharitonov, Jade Copet, Yossi Adi, Wei-Ning Hsu, Ali Elkahky, Paden Tomasello, Robin Algayres, Benoît Sagot, Abdelrahman Mohamed, et al. Generative spoken dialogue language modeling. *Transactions of the Association for Computational Linguistics*, 11: 250–266, 2023.

OpenAI. Developer notes on the realtime api. https://developers.openai.com/blog/realtime-api, 2025. Accessed 2026-09-23.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, 2024.

oto.earth. otoSpeech-full-duplex-task-oriented-20h: Full-duplex task-oriented conversations. https://huggingface.co/datasets/otoearth/otoSpeech-full-duplex-task-oriented-20h, 2026. License: CC BY 4.0.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Junyoung Park, Dalton Jones, Matthew Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. KeyDiff: Key similarity-based KV cache eviction for long-context LLM inference in resource-constrained environments. *Advances in Neural Information Processing Systems*, 38:5983–6019, 2026.

Yizhou Peng, Yi-Wen Chao, Dianwen Ng, Yukun Ma, Chongjia Ni, Bin Ma, and Eng Siong Chng. FD-Bench: A Full-Duplex Benchmarking Pipeline Designed for Full Duplex Spoken Dialogue Systems. In *Interspeech 2025*, pp. 176–180, 2025. doi: 10.21437/Interspeech.2025-739.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. CAKE: Cascading and adaptive KV cache eviction with layer preferences. In *International Conference on Learning Representations*, volume 2025, pp. 89945–89977, 2025.

Rajarshi Roy, Jonathan Raiman, Sang-gil Lee, Teodor-Dumitru Ene, Robert Kirby, Sungwon Kim, Jaehyeon Kim, and Bryan Catanzaro. PersonaPlex: Voice and role control for full duplex conversational speech models. In *ICASSP 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 16137–16141. IEEE, 2026.

Jiafei Song, Fengwei Zhou, Jin Qu, Wenjin Jason Li, Tong Wu, Gengjian Xue, Zhikang Zhao, Daomin Wei, Yichao Lu, and Bailin Na. EvoComp: Learning visual token compression for multimodal large language models via semantic-guided evolutionary labeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3532–3542, June 2026.

Yuxuan Tian, Zihan Wang, Yebo Peng, Aomufei Yuan, Zhiming Wang, Bairen Yi, Xin Liu, Yong Cui, and Tong Yang. KeepKV: Achieving periodic lossless KV cache compression for efficient LLM inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33259–33267, 2026.

Bandhav Veluri, Benjamin N Peloquin, Bokai Yu, Hongyu Gong, and Shyamnath Gollakota. Beyond turn-based interfaces: Synchronous LLMs as full-duplex dialogue agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21390–21402, 2024.

Heng Wang, Jielin Qiu, Wenting Zhao, Cheng Qian, Liangwei Yang, Jiawei Han, Heng Ji, Silvio Savarese, Shelby Heinecke, and Huan Wang. Random attention: Rethinking KV cache eviction for efficient reasoning. *arXiv preprint arXiv:2609.03430*, 2026a.

Peng Wang, Songshuo Lu, Yaohua Tang, Sijie Yan, Wei Xia, and Yuanjun Xiong. A full-duplex speech dialogue scheme based on large language model. In *Advances in Neural Information Processing Systems*, volume 37, pp. 13372–13403, 2024a.

Shibo Wang, Zicheng Zhang, Libo Wang, and Junfeng Ma. PACE: A playback-aligned context engine for LLM-based full-duplex voice dialogue. *arXiv preprint arXiv:2608.07631*, 2026b.

Xiong Wang, Yangze Li, Chaoyou Fu, Yike Zhang, Yunhang Shen, Lei Xie, Ke Li, Xing Sun, and Long Ma. Freeze-Omni: A smart and low latency speech-to-speech dialogue model with frozen LLM. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63345–63354. PMLR, 13–19 Jul 2025.

Yuxuan Wang, Peize He, Xiyan Gui, Xiaoqian Liu, Junhao He, Xuyang Liu, Xuming Hu, and Linfeng Zhang. AudioKV: KV cache eviction in efficient large audio language models. *arXiv preprint arXiv:2604.06694*, 2026c.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive KV cache merging for LLMs on long-context tasks. *arXiv preprint arXiv:2407.08454*, 2024b.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. PyramidInfer: Pyramid KV cache compression for high-throughput LLM inference. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3258–3270, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.195. URL https://aclanthology.org/2024.findings-acl.195/.

Wenyi Yu, Siyin Wang, Xiaoyu Yang, Xianzhao Chen, Xiaohai Tian, Jun Zhang, Guangzhi Sun, Lu Lu, Yuxuan Wang, and Chao Zhang. SALMONN-omni: A standalone speech LLM without codec injection for full-duplex conversation. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=AsRB5nmlOD.

Qinglin Zhang, Luyao Cheng, Chong Deng, Qian Chen, Wen Wang, Siqi Zheng, Jiaqing Liu, Hai Yu, Chao-Hong Tan, Zhihao Du, et al. OmniFlatten: An end-to-end gpt model for seamless voice conversation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14570–14580, 2025.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient LLMs inference. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

Enshuai Zhou, Yifan Hao, Chao Wang, Rui Zhang, Di Huang, Jiaming Guo, Xing Hu, Zidong Du, Qi Guo, and Yunji Chen. LKV: End-to-end learning of head-wise budgets and token selection for LLM KV cache eviction. *arXiv preprint arXiv:2605.06676*, 2026.

A Related work

A.1 KV cache eviction

A transformer decoder caches the key and value of every past token. This cache occupies accelerator memory beside the model weights and grows with both the length of each sequence and the number of sequences served together (Zhang et al., 2023). Longer inputs (Li et al., 2024) and extended multi-round dialogue (Xiao et al., 2024) enlarge it further. In reasoning models that spend additional compute on longer generations, the cost of generation is determined by the size of this cache, not by how many tokens are produced (Łańcucki et al., 2025). KV cache eviction bounds the cache by retaining only a subset of its tokens under a fixed budget.

Most eviction methods rank cached tokens by a statistic derived from attention. These methods differ in the queries over which this statistic is accumulated, all past queries in H2O (Zhang et al., 2023), an observation window at the end of the prompt in SnapKV (Li et al., 2024), and the current query only in TOVA (Oren et al., 2024). CriticalKV (Feng et al., 2026b) adds the value states to the statistic and scores each entry by how much its removal perturbs the attention output. StreamingLLM (Xiao et al., 2024) sets the statistic aside and keeps tokens by position, retaining the initial ones that act as attention sinks and a window of recent ones. Other methods keep an attention score and change how the budget is divided. These methods assign more of the budget to lower layers (Cai et al., 2025b), to layers chosen by the dispersion and temporal shift of their attention (Qin et al., 2025), or to individual heads within a layer (Feng et al., 2025). Where attention is unavailable at decision time, the statistic is replaced by a proxy computed solely from the keys (Park et al., 2026) or by an estimate of the attention that future queries will assign (Devoto et al., 2025). Recent analyses question both how reliable and how decisive this statistic is. The assumption that the same entries stay important throughout generation has proven fragile (Feng et al., 2026a). Attention-based policies retain the attention sink yet lose other structurally important tokens, and without explicit protection of those tokens their quality collapses (Garcia, 2026). Once the prompt is protected, random per-head eviction performs on par with the best score-based evictors on reasoning tasks (Wang et al., 2026a). The tokens a method protects may therefore matter more than the order in which it ranks the rest. We examine which tokens a full-duplex cache must protect in Section 3.

Learned approaches train the eviction decision rather than derive it from attention, and differ mainly in the signal that trains the decision. DMC (Nawrot et al., 2024) and DMS (Łańcucki et al., 2025) pair a loss on output quality with a penalty that drives the compression ratio toward a target. DMC decides, per position and per KV head, whether to append a token or merge it into the preceding one. DMS turns the same decision into an eviction that takes effect only after a fixed number of steps. TrimKV (Bui et al., 2026b) instead bounds the cache by an absolute budget of M tokens and learns a per-token retention value from the contextual embedding while the backbone stays frozen. A successor of TrimKV (Bui et al., 2026a) calibrates these values across layers and heads, so that one global ranking allocates the budget. Rather than penalizing the amount retained, LOCRET (Huang et al., 2024) and LookaheadKV (Ahn et al., 2026) regress the score onto attention, the maximum pre-softmax attention from answer tokens in LOCRET and the importance estimated from the model's own response in LookaheadKV. ForesightKV (Dong et al., 2026) and KVpop (Hauzenberger et al., 2026) take their targets from future attention instead, the former by distilling an oracle eviction and the latter by predicting the attention a token will receive. DistillCache (Althoubi, 2026) and LKV (Zhou et al., 2026) dispense with attention targets and train the policy to preserve the output of the full model, through a reward based on KL divergence and through self-distillation from a frozen teacher. EvoComp (Song et al., 2026) carries a related idea to visual tokens, labeling the tokens to keep by an evolutionary search for subsets that minimize the output loss. In all of these methods, the value of a token is measured through attention or through the output distribution of the model. JET instead trains the eviction decision on a judgment of the reply that the model generates.

A.2 Full-duplex spoken dialogue models

A full-duplex model processes incoming user audio while generating its own output. In the frame-synchronous designs considered here, the streams of both speakers enter a single cache frame by frame. The cache therefore grows with the duration of the session, including intervals in which neither the user nor the model speaks. Full-duplex systems include dGSLM (Nguyen et al., 2023), Moshi (Défossez et al., 2024), SyncLLM (Veluri et al., 2024), OmniFlatten (Zhang et al.,

2025), Freeze-Omni (Wang et al., 2025), MinMo (Chen et al., 2025), VITA-1.5 (Fu et al., 2025), SALMONN-omni (Yu et al., 2025), PersonaPlex (Roy et al., 2026), MiniCPM-o 4.5 (Cui et al., 2026), Raon-SpeechChat (Kim et al., 2026), and the scheme of Wang et al. (2024a). Their turn-taking and interruption behavior is evaluated on dedicated benchmarks (Lin et al., 2025; 2026c;b;a; Peng et al., 2025; Jiang et al., 2026), and the field is reviewed by Chen & Yu (2025). Existing full-duplex systems manage this cache for serving and for correctness after interruptions. Voxtral Realtime (Liu et al., 2026) serves an encoder cache and a decoder cache, which advance at different rates, through a single paged-attention backend. Freeze-Omni (Wang et al., 2025) stores the cache per user so that any instance in a model pool can serve any session. PACE (Wang et al., 2026b) repairs the model-facing context after an interruption so that it contains only output already played to the user. Moshi (Défossez et al., 2024) identifies model size as the bottleneck for serving many users and applies post-training quantization to weights and activations, which all sessions share. The KV state that grows with each session, which bounds the number of sessions a server can hold, is the subject of this work.

A.3 KV CACHE COMPRESSION BEYOND EVICTION

KV cache memory has been reduced along several axes. Depth-wise sharing reuses a single cache across several layers (Filippova et al., 2026), and quantization lowers the numerical precision of each entry (Liu et al., 2024). Merging combines entries instead of discarding them (Zhang et al., 2024; Wang et al., 2024b; Tian et al., 2026). Filippova et al. (2026) place eviction on the temporal axis and regard depth-wise sharing as orthogonal to it. Our method operates on the temporal axis under a fixed budget of retained tokens per KV head. Retained tokens keep their original positions, so at a fixed frame rate the position of a token still encodes the time at which it was written. Because merging would combine tokens written at different times into a single entry, we evict tokens rather than merge them.

B BACKBONES

Full-duplex models differ in how the speech of the two speakers reaches the cache of the language model (Figure 5). We borrow the terms multi-stream and single-stream from Chen & Yu (2025) for whether the streams share a cache position, and divide single-stream models by where the agent's speech is generated. In Moshi (Défossez et al., 2024), a multi-stream model, the embedding of the generated codec tokens is summed with the user input, so every cache position holds both speakers. Raon-SpeechChat (Kim et al., 2026) writes the embedding of its own codec tokens to a position of its own, whereas MiniCPM-o 4.5 (Cui et al., 2026) and Freeze-Omni (Wang et al., 2025) generate speech with a separate speech decoder from the hidden states of the LLM and write no agent audio to the cache. Eviction by token type requires that each speaker have its own positions, and our two backbones cover both single-stream types. Table 2 lists the module of each model behind each block, and Figure 6 draws the two backbones in more detail. Table 3 gives the tokens that one step of each backbone writes to the cache.

Table 2: Modules of each model that correspond to the blocks of Figure 5.

Block	Moshi	Raon-SpeechChat	MiniCPM-o 4.5
Speech encoder	Mimi encoder	causal speech encoder and adaptor	Whisper encoder and MLP projector
LLM	Temporal Transformer	Qwen3-VL-8B	Qwen3-8B
Codec generator	Depth Transformer	speech generation expert and residual code predictor	–
Codec embedding	sum of codebook embeddings	sum of 16 codebook embeddings and adaptor	–
Speech decoder	–	–	speech decoder and flow-matching decoder

Token types. Agent text contains the agent's text tokens, which correspond to the text stream of Chen & Yu (2025). This partition is our refinement of the formalism of Section 2. User audio refines $S^{\mathcal{E}}$, while agent audio, agent text and turn control refine $S^{\mathcal{A}}$.

Tokens of each step. MiniCPM-o 4.5 (Cui et al., 2026) is an omni-modal model that supports visual input in addition to audio. In this work, only audio input is used, so no visual tokens enter the cache. MiniCPM-o 4.5 encodes input audio with a Whisper encoder at 50 features per second. An MLP projector then compresses these features by a factor of five, so that the language model receives

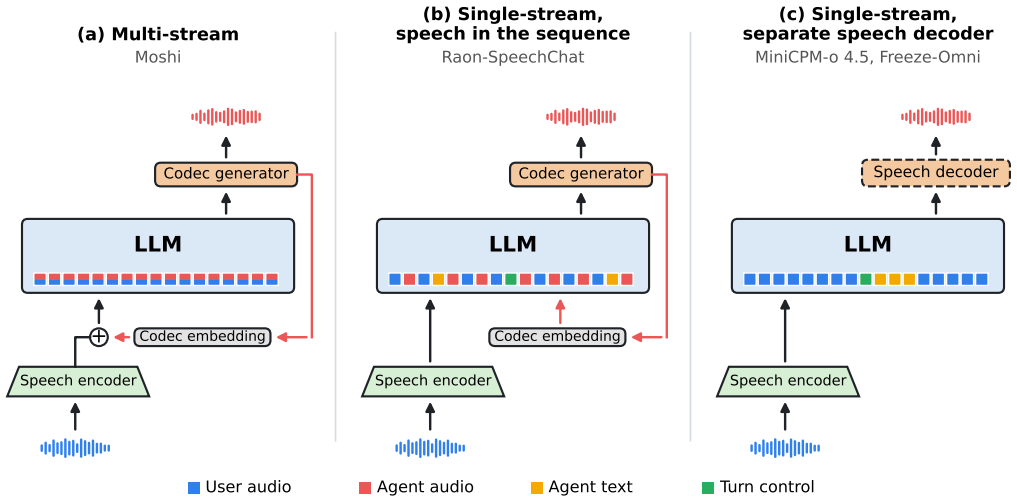

Figure 5: Diagrams of three types of full-duplex model. The row inside each LLM is its cache, colored by token type.

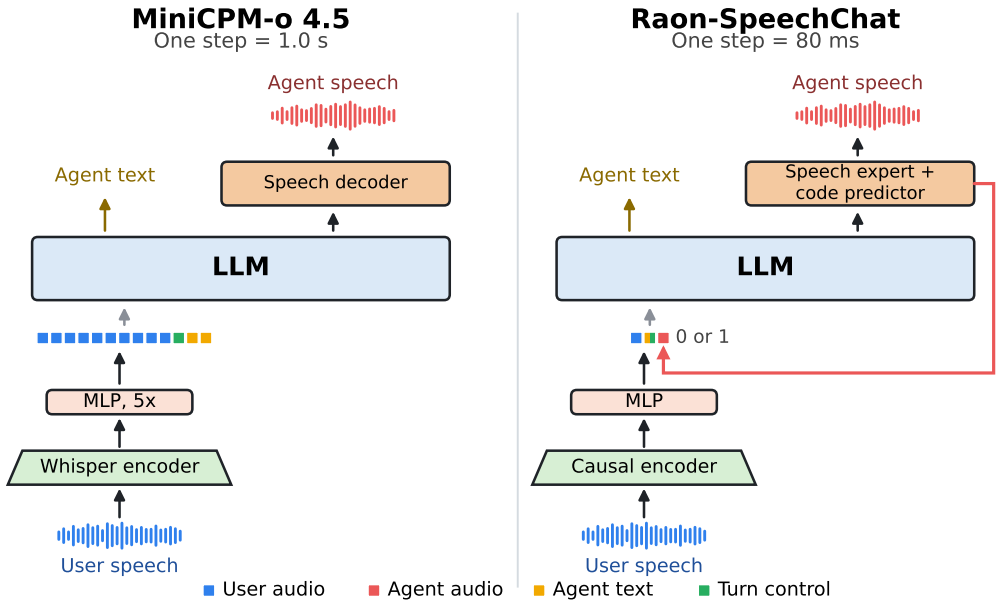

Figure 6: Diagrams of MiniCPM-o 4.5 and Raon-SpeechChat. The row under each LLM holds the tokens that one step writes to the cache.

ten audio tokens for each second of input. The language model generates text, and a separate speech decoder produces speech tokens from its hidden states outside the cached sequence. Agent audio therefore occupies no position in the cache. Each one-second input chunk is enclosed by a pair of chunk markers. Within a chunk the model first predicts a turn-control token that selects speaking or listening, and closes the output with a stop token.

Raon-SpeechChat (Kim et al., 2026) encodes user speech with a causal speech encoder and generates speech through the Mimi codec at 12.5 Hz, that is, one frame every 80 ms. Each frame is described by sixteen codec tokens, one semantic and fifteen acoustic, which are summed into a sin-

Table 3: Tokens written to the cache in one step of each backbone. Agent text and turn control vary with the content of the step.

	MiniCPM-o 4.5	Raon-SpeechChat
Step duration	1.0 s	80 ms
User audio	10	1
Agent audio	–	1
Agent text	variable	0 or 1
Turn control	1 control, 2 chunk markers, 0 or 1 stop token	0 or 1

gle embedding that re-enters the backbone. Each frame therefore contributes one agent audio token alongside one user audio token. Agent text and speech tokens are interleaved in the same sequence and aligned word by word. Transitions between listening and speaking are governed by special tokens. In the released implementation, the end-of-output padding token and the backchannel token are appended to the sequence and are cached, whereas the silence token is predicted at every listening frame without being appended. The step durations of the two backbones differ by a factor of 12.5, but the lower bounds of $13n$ and $25n$ tokens for a session of n seconds in Section 2 differ by less than a factor of two, because MiniCPM-o 4.5 writes ten audio tokens into each one-second step.

C GRADIENTS THROUGH THE BUDGET MASK

Each scored token j carries one logit $\log \alpha_j$, and the two terms of the objective reach it through different paths.

Language-modeling term. The retained set of Eq. (5) is piecewise constant in $\log \alpha$, so its derivative is zero almost everywhere. Relaxing it in the forward pass, by weighting keys with $\sigma(\log \alpha_j)$ or with sampled hard concrete gates, would let evicted keys contribute to attention and would not hold the cache at M tokens, so the loss would be measured on a cache that inference never holds. We therefore keep the hard rule in the forward pass and replace its derivative in the backward pass,

$$\tilde{z}_{qj} = \sigma(\log \alpha_j) + \mathrm{sg}\left(z_{qj} - \sigma(\log \alpha_j)\right), \qquad u_{qj} = \frac{\mathbf{q}_q^\top \mathbf{k}_j}{\sqrt{d_k}} + \log \max(\tilde{z}_{qj}, \epsilon), \tag{16}$$

where $z_{qj} = \mathbf{1}[j \in \mathcal{K}_t]$ for the step t of query q, sg stops the gradient, u_{qj} is the attention logit in every layer and head, d_k is the head dimension, and $\epsilon = 10^{-6}$. The forward pass applies the budget mask, with evicted keys scaled by ϵ, and the loss reaches $\log \alpha_j$ through $\sigma'(\log \alpha_j)$ for every key that the mask retains. Evicted keys receive no gradient from this term, and keys of the prefix and of the current and preceding steps enter as the constant $\tilde{z}_{qj} = 1$.

L_0 term. The open probability p_j of Section 4.3 is a smooth function of $\log \alpha_j$ for every scored token, retained or evicted, so this term reaches all logits. It lowers the logits of tokens that the language-modeling term does not support, including those already evicted, and both terms act on $\log \alpha_j$ through the same logistic function.

Gradient paths. The gate reads hidden states of the Full KV cache pass, which carries no gradient, so the gradient with respect to $\log \alpha$ updates only $\boldsymbol{\phi}$, and the backbone is updated by the language-modeling term of the masked pass.

Mask from the gate logit. A natural alternative is to build the training mask from the combined score of Eq. (10), so that training evicts the tokens that inference evicts. We rank by the gate logit instead, because the gate is the only channel trained to score tokens. Attention and recency have no parameters of their own and are computed from the cache, so any cost of an eviction that the loss measures can enter the policy only through the gate. For the gate to learn this cost, the loss has to evaluate evictions that the gate decided. Under the combined rule most evictions would be decided by attention and recency, yet the change in loss they cause would still be passed to the gate logits by the straight-through estimator, scaled by w_g. The gate would then be fit to correct the two statistics at the weights used in training, not to rank tokens by the cost of losing them, and this

correction would matter least at small budgets, where the choice among tokens that the statistics do not separate decides the reply. With a mask determined by the gate logit, every eviction during training is a decision of the gate, so the loss evaluates its ranking directly, and the statistics are added only at inference.

Training step. Algorithm 1 gives one training step of Section 4.3, and Figure 7 shows the budget mask of Eq. (5) on both backbones.

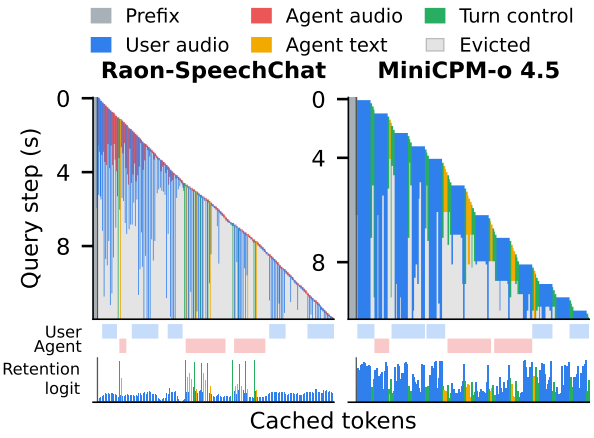

Figure 7: Budget mask at $M = 64$ over the first 12 s of a Behavior-SD training session. Rows are query steps and columns are cached tokens, gray for evicted tokens. Below each panel are the speaking intervals and the retention logit $\log \alpha_j$.

Algorithm 1 Training step of DuplexKV

Require: session x, backbone weights θ, gate g_ϕ, budget M, coefficient λ

$\{\mathbf{h}_j\} \leftarrow \mathrm{sg}\left(f_\theta(x)\right)$ ▷ First pass (Full KV cache)
$\log \alpha_j \leftarrow g_\phi(\mathbf{h}_j)$ for every token j ▷ Eq. (4)
$\mathcal{K}_0 \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  $\mathcal{K}_t \leftarrow \mathrm{Top}_M(\mathcal{K}_{t-1} \cup \mathcal{N}_t; \log \alpha)$ ▷ Eq. (5)
  $\mathcal{R}_i \leftarrow \mathcal{P} \cup \mathcal{K}_t \cup \mathcal{N}_{t-1} \cup \mathcal{N}_t$ for $i \in \mathcal{N}_t$ ▷ Steps $t-1$, t exempt from the budget
end for
$\mathcal{L}_{\mathrm{LM}} \leftarrow$ Eq. (6) on $f_\theta(x; \mathcal{R})$ ▷ Second pass, straight-through
$\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{LM}} + \lambda \mathcal{L}_0$ ▷ Eq. (9)
update θ and ϕ with $\nabla \mathcal{L}$

D Data

D.1 Training data

We train on Behavior-SD (Lee et al., 2025), a corpus of two-channel spoken dialogues with word-level timestamps for each channel. One channel is fed to the backbone as the user stream and the other as the agent stream. The training split contains 105,178 dialogues, and the validation split contains 932 dialogues that share no transcript with the training split. Each training example is a 60 s window of a dialogue, the same for both backbones, and each training step uses one window per device.

D.2 Evaluation sets

FDB v1.0 user interruption. The user interruption task of Full-Duplex-Bench v1.0 (Lin et al., 2025) contains 200 episodes. In each episode the user speaks a context turn, the model begins to

answer, and the user interrupts it with a new turn. The takeover rate (TOR) is the fraction of episodes in which the model takes the turn after the interruption, and a judge rates the reply that follows it on a scale from 0 to 5 (Appendix E).

Behavior-SD user interruption. To obtain a second evaluation set that shares no episode with FDB, we build 185 interruption episodes from the validation split of Behavior-SD in the episode layout of FDB. Each episode consists of the user turn that the model was answering, the interval in which the model spoke before it was cut off, the interrupting user turn, and a 15 s interval for the reply. The interval before the interruption is taken from the dialogue and clipped to the range that FDB covers, and events that are not preceded by a user turn followed by a model turn are discarded. The episodes are scored with the same judge prompt and the same metrics as FDB.

otoSpeech long sessions. otoSpeech-full-duplex-task-oriented-20h contains 58 English two-speaker task-oriented dialogues, 20.0 hours in total, with sessions of 11.1 to 24.9 minutes recorded with one channel per speaker. We transcribe each channel with Parakeet-TDT-0.6B-v2 and take as a boundary the end of a user utterance of at least 2 s that the other speaker answers after a pause of 0.2 to 3 s, keeping only boundaries after which the user does not speak again within 2 s. At each of 1, 4, 8, 10, 12, 14, 16, 20 and 24 minutes we select 30 boundaries within 60 s of that time, or within 90 s at 24 minutes, where fewer sessions leave the 30 s needed for the reply, at most one per session where possible and at least 60 s apart within a session, which gives 270 points. Up to the boundary the model receives the user channel and is forced to emit the words of the recorded speaker in its role, and after the boundary it receives silence and generates its own reply for 30 s. The judge reads the last 12 turns before the request, the request and the reply, so that the context it reads does not depend on the session time. Figure 8 shows the takeover rate of the Full KV cache of Raon-SpeechChat with the released weights. Raon-SpeechChat takes 30% of the turns at 1 minute and at most 3% from 4 minutes on, which leaves too few replies to compare eviction policies.

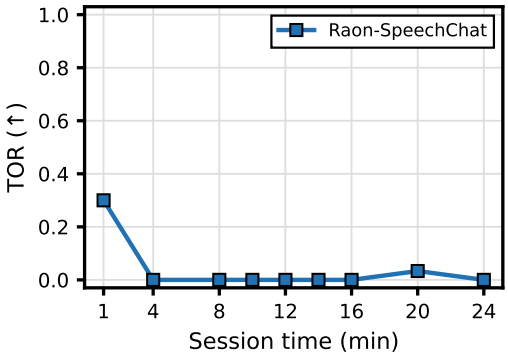

Figure 8: TOR of the Full KV cache with the released weights on otoSpeech sessions.

E Evaluation protocol

TOR and latency are computed by the scorer released with FDB, which counts a takeover when the reply after the interruption spans at least 1 s or contains more than three words. The judge rates a reply only when the model took the turn, as in the released evaluation, so we report the mean judge score over those episodes together with their number, n. A paired comparison between two methods uses the episodes in which both took the turn, and its 95% confidence interval comes from a paired bootstrap over those episodes. The judge prompt is the released one, reproduced in Figure 10. We replace the released judge model with gpt-5.6-luna with reasoning effort set to none and keep the prompt, the parser and the gating on TOR unchanged. The judge is called with the released seed of 0 and the default temperature, and a completion that cannot be parsed is requested again up to three times.

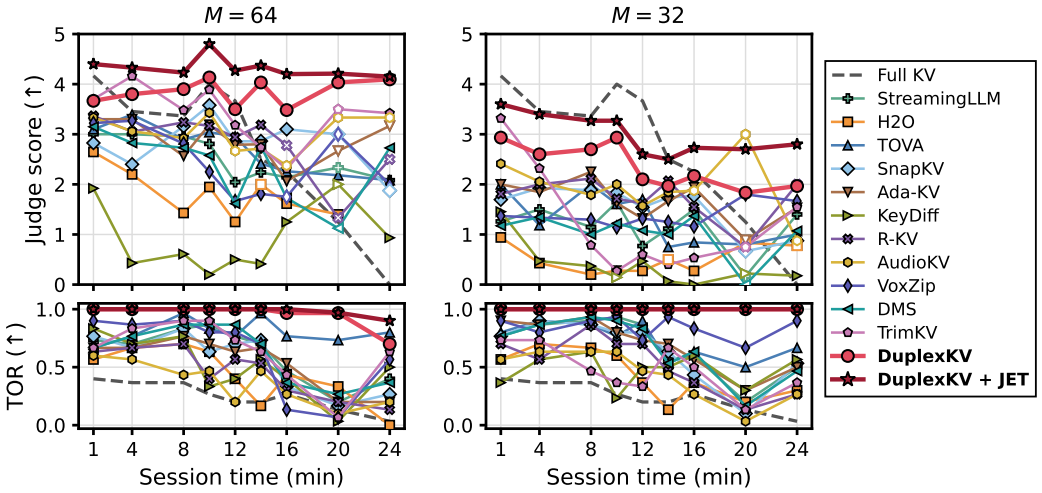

Figure 9: Comparison with eviction policies at each session time over otoSpeech sessions with MiniCPM-o 4.5. Hollow markers denote means over fewer than 10 replies.

E.1 Agreement with a second judge

To check that the judge scores do not depend on the choice of judge, we rescore the Full KV cache and DuplexKV at $M = 32$ on both backbones, 768 rated replies in total, with gpt-6-astra at medium reasoning effort, keeping the prompt, the parser and the gating on TOR unchanged. Over all replies the quadratic weighted kappa is 0.885, and 98.3% of the scores differ by at most one point. The second judge orders DuplexKV and the Full KV cache in the same direction as the primary judge on both backbones.

F Judged eviction tuning

Algorithm 2 gives the procedure of Section 4.5, and Figure 11 follows the collection of verdicts over the turns of one dialogue. Restricting the judged tokens to $\mathcal{J}$ of Eq. (13) also bounds the number of judgments per reply.

Algorithm 2 Judged eviction tuning (JET)

Require: gate g_ϕ trained with Eq. (9), frozen backbone θ, channel weights (w_g, w_a, w_d), budget M, recorded dialogues $\mathcal{D}$, judge G, coefficients λ, η

$\mathcal{B} \leftarrow \emptyset$
for all dialogues $x \in \mathcal{D}$ and agent turns r of x **do**
  $\mathcal{K} \leftarrow$ Full KV cache of the recorded $x_{<r}$
  generate $\hat{a}_r$ under Eqs. (10) and (11) at budget M ▷ Eviction only within turn r
  $\bar{z}_j \leftarrow$ Eq. (12) at $t^\star$
  $\mathcal{J}_r \leftarrow \{\, j \in \mathcal{U}_r : 0 < \bar{z}_j < 1 \,\}$ ▷ Eq. (13)
  group $\mathcal{J}_r$ into word items $\mathcal{W}_r$
  $y_w \leftarrow G\big(x_{<r},\, \hat{a}_r,\, \mathcal{W}_r\big)$ for every $w \in \mathcal{W}_r$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{\, (j,\, y_{w(j)}) : j \in \mathcal{J}_r \,\}$
end for
repeat
  $\mathcal{L} \leftarrow$ Eq. (15) with Algorithm 1, θ frozen
  update ϕ with $\nabla\mathcal{L}$
until the step budget is spent

Judge prompt for FDB user interruption (system message)

```
The scenario is that the user and AI are talking in the spoken
conversation.
The user first speaks, then the AI responds.  But when AI is speaking,
the user interrupts the AI's turn.
Your task is to rate the quality of AI's response after the user
interrupt the turn.

Below is the rating guideline (from 0 to 5, 0 is the worst and 5 is
the best):
- 0:  The AI's response is totally unrelated to the user's
interrupting turn.
- 1:  The AI's response is not related to the user's interrupting
turn.
- 2:  The AI's response is slightly related to the user's interrupting
turn.
- 3:  The AI's response is related to the user's interrupting turn.
- 4:  The AI's response is highly related to the user's interrupting
turn.
- 5:  The AI's response is perfectly related to the user's
interrupting turn.

Firstly, briefly analyze the user's interrupting turn and the AI's
response
Then, you must return the overall output as the following format:
Analysis:  [Your analysis].
I would rate the AI's response as [Rating].
```

Judge prompt for FDB user interruption (user message)

```
- Contextual user turn:  {in_before_interrupt_text}
- User interrupting turn:  {in_interrupt_text}
- AI's response:  {out_after_interrupt_text}
```

Figure 10: Judge prompt of the FDB v1.0 user interruption task, reproduced verbatim from the released evaluation code. Placeholders are filled with the transcripts of the episode.

F.1 Judge prompt and items

The JET judge is Qwen3.8-27B run locally with vLLM, with thinking disabled and the default sampling configuration of the model. Each request is a single user message that contains the document of Figure 12, and there is no system message. The reply must contain a JSON object that assigns keep or discard to every item, and a reply that cannot be parsed is requested again with the next seed. The dialogue lists the recorded turns up to the judged turn, and the reply that the model generated under the eviction policy follows under its own heading on one line. Each candidate token is assigned to the spoken word whose time span it overlaps most, and all tokens under one word form one item that receives one verdict. A dialogue with several judged turns yields one training record. Where the windows of consecutive turns overlap, labels on agent text are taken from the later turn and labels on user audio from the earlier turn.

F.2 Verdict collection

Verdicts are collected on 1,000 Behavior-SD training dialogues per backbone, rolled out with the trained gate and backbone under the eviction policy at $M = 32$. This gives 5,274 judged turns with 159,890 items on Raon-SpeechChat and 5,200 turns with 128,808 items on MiniCPM-o 4.5, of which 20.6% and 25.1% are judged keep.

F.3 Consistency of the verdicts

We check that the JET judge chooses items by their content rather than by their position in the list. We draw 1,000 documents from Raon-SpeechChat at random and judge each again with a different

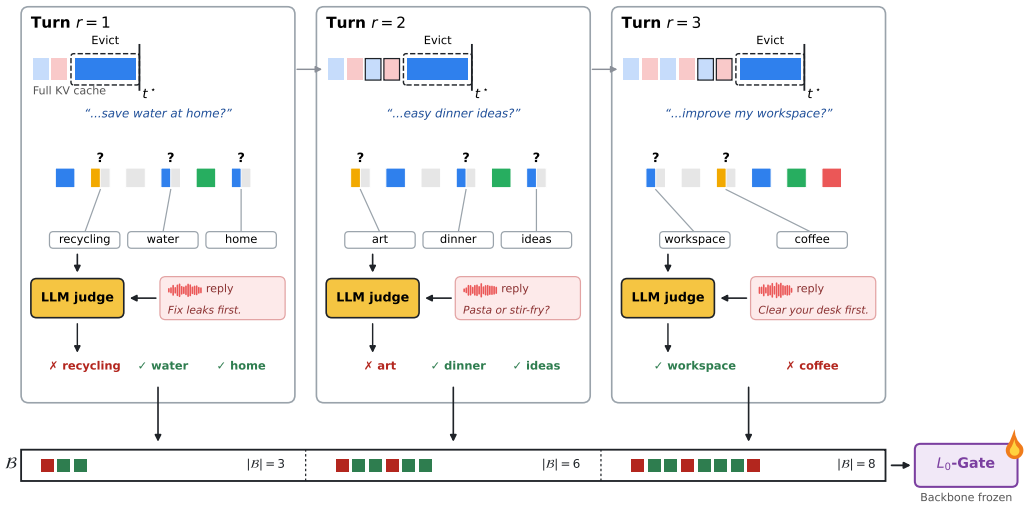

Figure 11: Verdict collection over the turns of one dialogue, from the eviction within each turn to the set $\mathcal{B}$ used to tune the L_0-Gate. Words and replies are illustrative.

seed (30,184 items), and we judge the 982 documents with at least four candidates again with the candidate list shuffled under two orders (30,138 items), comparing the verdicts item by item with the original ones (Table 4). Shuffling the list changes the verdicts no more than judging the same document again, with an agreement of 81.4% and a Gwet's AC1 of 0.744 against 79.2% and 0.690, so the verdicts follow the items rather than their positions. We report AC1, which is robust to the low share of keep verdicts. The judge document gives the dialogue, the reply and the items, and does not state whether an item is held in the cache, so that each verdict rests on the content of the item.

Table 4: Per-item agreement between the JET verdicts and the verdicts of a repeated run.

Comparison	Documents	Items	Agreement	Gwet's AC1
Same document, another seed	1,000	30,184	79.2%	0.690
Candidate list shuffled, mean of two orders	982	30,138	81.4%	0.744

F.4 Judge score before and after JET

Figure 13 compares the judge score of DuplexKV before and after JET at $M = 64$ and $M = 32$ on both backbones.

F.5 Retained words before the reply

To locate the change that JET makes to the policy, we compare the retained sets of DuplexKV and DuplexKV + JET at the step at which the reply begins. On Raon-SpeechChat, the share of KV heads that retain the agent's own earlier content words at the reply step falls from 7.8% to 3.5% at $M = 32$ and from 21.1% to 11.1% at $M = 64$. In the episodes whose score JET raises, the reply reuses a content word of the earlier topic less often, 19% against 47%, and contains a word of the request more often, 14% against 6% (Figure 14).

G Implementation details

The gate reads the last-layer representation of each token, applies an RMSNorm and one linear projection, and is shared across layers and KV heads. With a hidden size of 4,096 on both backbones it has 8,193 parameters. The bias of the projection is initialized so that every token is retained at

Judge prompt for JET

```
You are an expert evaluator assessing dialogue memory retention for a
real-time conversational AI.
The model holds a strictly limited working memory.  Several items from
what the user said were on the boundary between being kept and being
forgotten when the model produced its last reply.

# Dialogue
[{t}s] {User|Model}:  {turn text}
...

# Last Reply
[{t}s] Model:  {reply text}

# Candidate Items
[{i}] {start}-{end}s {User|Model}:  {word}
...

# Task
For each candidate item, decide what should have happened to it.
  keep:  losing this item would have damaged the last reply
  discard:  the last reply would have been just as good without it
Judge each item on its own.  There is no quota:  mark as many keep or
discard as the dialogue warrants.  Every item must get one of the two.

# Output Format
Return a valid JSON object.  Every candidate ID from 1 to {n} must
appear exactly once, each with keep or discard.
{"verdicts":  {"1":  "keep", "2":  "discard"}}
```

Example input (dialogue and items) and output

```
# Dialogue
[  0.0s] User:  Hey you look a bit lost
[  2.0s] Model:  Oh
[  2.3s] User:  Can I help you with directions Okay

# Last Reply
[  8.0s] Model:  I'll show you the way It's not that far You just
need to turn left at the next corner and then follow the main road
for about two blocks The building will be on your right

# Candidate Items
[1]   2.0-3.0s Model:  Oh
[2]   2.6-2.8s User:  Can
[3]   2.8-2.9s User:  I
[4]   2.9-3.2s User:  help
[5]   3.2-3.4s User:  you
[6]   3.4-3.5s User:  with
[7]   3.6-4.3s User:  directions
[8]   6.4-7.0s User:  Okay

Output:  {"verdicts":  {"1":  "discard", "2":  "keep", "3":  "keep",
"4":  "keep", "5":  "keep", "6":  "keep", "7":  "keep", "8":
"discard"}}
```

Figure 12: Document given to the JET judge, with placeholders, and one input and output from MiniCPM-o 4.5. In the example, the header and task text shown above are omitted, and the line under Last Reply is the reply generated under the eviction policy.

the start of training. Training updates the language model of each backbone together with the gate, and the audio encoder and the speech decoder stay frozen. On Raon-SpeechChat the loss weight is 0.1 for padding and silence tokens and for the acoustic codes, and 1.0 for every other token, including the end-of-output padding and backchannel tokens. On MiniCPM-o 4.5 all weights are 1 and Z in Eq. (6) is the number of agent output tokens, and on Raon-SpeechChat Z is the number of positions in the batch. During training, the tokens written in the current and preceding steps are

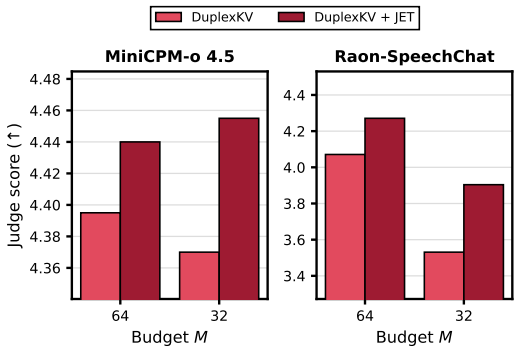

Figure 13: Comparison of judge score of DuplexKV before and after JET on the FDB v1.0 user interruption task.

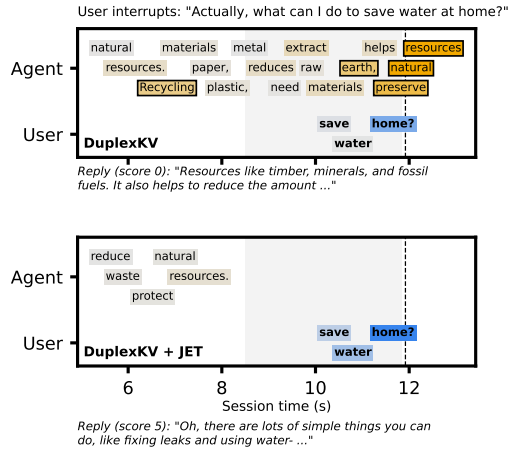

Figure 14: Words retained before the reply by DuplexKV and DuplexKV + JET in one episode of FDB v1.0 user interruption (Raon-SpeechChat at $M = 32$). Shading gives the share of KV heads that retain each word when the reply starts, and outlines mark shares of at least 0.3.

always visible to a query. On MiniCPM-o 4.5 a step is a one-second chunk that the model receives incrementally, first the chunk marker and the audio block and then one token at a time, and the budget rule is applied after each input. On Raon-SpeechChat a step is one 80 ms frame. Table 5 lists the remaining settings. We select $\lambda = 0.05$ with inference weights 0.2:1:1 for MiniCPM-o 4.5 and $\lambda = 0.01$ with 1:2:3 for Raon-SpeechChat on the interruption episodes built from the validation split of Behavior-SD. At inference, text is decoded greedily on both backbones, and the speech tokens of MiniCPM-o 4.5 are sampled by its speech decoder at temperature 0.8 with a fixed seed. Training and evaluation use nodes of 8 NVIDIA A100 80GB GPUs, one node for MiniCPM-o 4.5 and two for Raon-SpeechChat.

Inference procedure. Algorithm 3 gives one step of eviction at inference with the eviction policy of Section 4.4. The gate scores each token once, after the forward pass that writes it, so a token becomes a candidate from the next eviction onward. On MiniCPM-o 4.5, whose step arrives as several inputs, eviction and scoring follow each input. The attention channel accumulates the attention that each cached token receives in each KV head. Evicted keys and values are removed from the cache of their head, and retained tokens keep their original positions.

H BASELINES

Each baseline follows the official code of its method, or its paper where no code is released, and is implemented in one framework inside the decoding loop of each backbone. Training-free baselines

Table 5: Training and inference settings.

	MiniCPM-o 4.5	Raon-SpeechChat
Steps	300	300
GPUs, sessions per GPU	8, 1	16, 1
Window length	60 s	60 s
Optimizer	AdamW, weight decay 0.01	AdamW, weight decay 0.01
Schedule	cosine, 5% warmup, backbone to 0.1 of peak	cosine, 5% warmup, backbone to 0.1 of peak
Learning rate, gate	3×10^{-4}	1×10^{-4}
Learning rate, backbone	1×10^{-5}	3×10^{-6}
Gradient clipping	1.0	1.0
Precision	bfloat16	bfloat16
Training budget M	64	64
λ	0.05	0.01
β, γ, ζ	$2/3, -0.1, 1.1$	$2/3, -0.1, 1.1$
Inference weights (w_g, w_a, w_d)	$0.2 : 1 : 1$	$1 : 2 : 3$
JET learning rate, gate	1×10^{-5}	3×10^{-4}
JET steps	300	400
JET judge-term weight η	1	1

Algorithm 3 Eviction step of DuplexKV at inference

Require: cache $\mathcal{K}_{t-1}^{(\ell,h)}$ for every layer ℓ and KV head h, new tokens $\mathcal{N}_t$, gate $g_{\boldsymbol{\phi}}$, weights (w_g, w_a, w_d), budget M

1: **for all** layers ℓ and KV heads h **do**
2: $\quad \mathcal{C} \leftarrow \mathcal{K}_{t-1}^{(\ell,h)}$ ▷ Candidates, the current step excluded
3: $\quad s_j \leftarrow w_g\, \Pi(\log \alpha)_j + w_a\, \Pi(A^{(\ell,h)})_j + w_d\, \Pi(d)_j$ for $j \in \mathcal{C}$ ▷ Eq. (10)
4: $\quad \mathcal{K}_t^{(\ell,h)} \leftarrow \mathcal{N}_t \cup \mathrm{Top}_{M-|\mathcal{N}_t|}(\mathcal{C};\, s)$ ▷ Eq. (11)
5: **end for**
6: $\{\mathbf{h}_j\}_{j\in\mathcal{N}_t},\ \Delta A \leftarrow f_\theta(\mathcal{N}_t;\, \mathcal{K}_t)$ ▷ Attention received by cached tokens
7: $A^{(\ell,h)} \leftarrow A^{(\ell,h)} + \Delta A^{(\ell,h)}$ for every ℓ and h
8: $\log \alpha_j \leftarrow g_{\boldsymbol{\phi}}(\mathbf{h}_j)$ for $j \in \mathcal{N}_t$

run on the released weights of each backbone. This section describes what we change so that every method runs on a streaming cache under the budget accounting of DuplexKV, and why. Table 6 gives the results of every baseline on FDB v1.0 user interruption, including those left out of Table 1.

Table 7 runs four baselines of Table 1 and the Full KV cache on the backbone trained jointly with the gate of DuplexKV, with every other setting as in Table 1.

Budget accounting. All methods keep the prefix outside the budget and count chunk markers and turn-control tokens toward M, as DuplexKV does, so that every method holds the same number of conversation tokens per KV head. Retained tokens keep their original positions. Methods that give every KV head the same budget remove evicted tokens from the cache. Methods whose budget differs across heads or layers mask evicted tokens instead, because the cache is compacted to one number of tokens for every head and layer, and a masked token receives zero attention weight as a removed one does.

Per-step scoring. Most methods were proposed for a prompt that is compressed once after prefill, or for a long generation compressed at fixed intervals. In a full-duplex session the cache grows at every step and the prompt has no end at which to compress, so each method applies its scoring rule at every decoding step to the tokens written so far and keeps the tokens it ranks highest within the budget. An evicted token is not restored. Methods that score tokens with the attention of the last w queries use the last w queries of the stream, including those of the current step, with $w = \min(32, M/2)$ unless Table 8 gives another window. LookaheadKV appends 32 learned lookahead queries after the prompt, and we append them after the tokens written so far at every step and read their attention without writing them to the cache.

DMS under a fixed budget. DMS evicts a token when its adapter outputs an eviction decision for it, after a delay window, so the number of tokens it keeps varies with the input and across heads. To compare at the same M, we average the eviction probability of the adapter over layers and heads

Table 6: Comparison with every eviction policy on FDB v1.0 user interruption at $M \in \{32, 64\}$, with the judge score averaged over the episodes in which the model takes the turn.

	MiniCPM-o 4.5				Raon-SpeechChat			
	$M = 32$		$M = 64$		$M = 32$		$M = 64$	
Method	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)
Full KV	4.03	0.96	4.03	0.96	4.01	0.94	4.01	0.94
Recency								
StreamingLLM (Xiao et al., 2024)	3.19	<u>0.99</u>	3.71	<u>0.99</u>	1.17	0.33	1.61	0.48
Attention								
H2O (Zhang et al., 2023)	0.43	0.95	3.84	0.96	0.88	0.89	2.10	<u>0.98</u>
Scissorhands (Liu et al., 2023)	2.72	0.98	4.20	0.96	1.53	0.15	3.33	0.41
PyramidInfer (Yang et al., 2024)	3.60	<u>0.99</u>	4.32	<u>0.99</u>	0.64†	0.06	3.48	0.27
TOVA (Oren et al., 2024)	3.36	0.97	4.25	0.95	0.45†	0.06	0.41†	0.09
SnapKV (Li et al., 2024)	3.74	<u>0.99</u>	4.08	0.97	2.00	0.56	4.10	0.83
DefensiveKV (Feng et al., 2026a)	4.13	0.98	4.02	0.98	3.07	0.30	<u>4.20</u>	0.40
LookaheadKV (Ahn et al., 2026)	4.08	0.97	4.23	0.97	3.65	0.86	4.05	0.88
Budget allocation								
CAKE (Qin et al., 2025)	3.76	0.96	4.05	0.97	2.55	0.49	3.78	0.74
PyramidKV (Cai et al., 2025b)	3.57	<u>0.99</u>	4.15	0.96	1.91	0.38	3.72	0.61
Ada-KV (Feng et al., 2025)	3.77	0.96	4.20	0.98	2.27	0.47	4.13	0.80
Key statistics								
R-KV (Cai et al., 2025a)	4.07	0.97	4.29	0.96	<u>3.78</u>	0.59	3.88	0.94
ChunkKV (Liu et al., 2025)	3.87	0.97	4.25	0.97	2.37	0.72	4.01	0.90
KeyDiff (Park et al., 2026)	1.22	0.88	2.52	0.96	0.44†	0.04	1.44	0.42
Audio								
AudioKV (Wang et al., 2026c)	4.29	0.96	4.31	0.98	2.01	0.42	4.12	0.80
VoxZip (Jia et al., 2026)	0.47	0.80	3.59	<u>0.99</u>	0.57	**0.98**	1.88	0.90
Learned								
Locret (Huang et al., 2024)	3.56	0.94	4.26	0.95	3.46	0.82	3.80	0.95
DMS (Łańcucki et al., 2025)	3.16	<u>0.99</u>	3.95	<u>0.99</u>	0.87	0.30	3.38	0.80
TrimKV (Bui et al., 2026b)	4.06	0.98	4.35	<u>0.99</u>	3.18	0.51	4.12	0.62
ForesightKV (Dong et al., 2026)	3.99	0.95	4.28	0.98	3.56	0.62	3.96	0.83
DuplexKV (Ours)	<u>4.37</u>	**1.00**	<u>4.39</u>	**1.00**	3.53	<u>0.97</u>	4.07	<u>0.98</u>
DuplexKV + JET (Ours)	**4.46**	**1.00**	**4.44**	**1.00**	**3.90**	**0.98**	**4.27**	**0.99**

Bold and underline mark the highest and second highest value among eviction methods in each column. † Fewer than 30 episodes with TOR = 1.

Table 7: Baselines on the jointly trained backbone of DuplexKV, FDB v1.0 user interruption at $M \in \{32, 64\}$.

	MiniCPM-o 4.5				Raon-SpeechChat			
	$M = 32$		$M = 64$		$M = 32$		$M = 64$	
Method	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)	Judge (↑)	TOR (↑)
Full KV	4.33	1.00	4.33	1.00	3.85	0.96	3.85	0.96
StreamingLLM	2.24	1.00	4.02	1.00	1.14	0.97	3.27	0.96
SnapKV	4.00	1.00	4.28	1.00	2.60	0.97	3.96	0.98
R-KV	4.07	1.00	4.40	1.00	3.10	0.97	3.35	0.99
AudioKV	4.36	1.00	4.37	1.00	1.98	0.98	3.73	0.99
DuplexKV (Ours)	4.37	1.00	4.39	1.00	3.53	0.97	4.07	0.98
DuplexKV + JET (Ours)	4.46	1.00	4.44	1.00	3.90	0.98	4.27	0.99

and keep the tokens with the lowest average. DMS does not evict a token within its delay window, so we reserve the newest $\max(48, M/2)$ tokens, capped at M, which covers the delay window of 48 tokens. The released delay window of 512 tokens exceeds every budget we evaluate, so we train the adapter with a window of 48.

Training of learned baselines. No learned method releases a checkpoint for our backbones, so we train each on the training split of Behavior-SD with the backbone weights frozen. DMS and TrimKV are trained through the language model with their own objectives, and Locret, ForesightKV and LookaheadKV are fitted offline on Full KV cache traces of Behavior-SD dialogues. LookaheadKV

also trains a rank-8 LoRA that is active only on its lookahead queries. Table 8 gives the venue, the importance score, the reserved tokens and the training of each method.

Table 8: Baselines with their venue, importance score, tokens reserved beyond the prefix, and the training of the learned methods.

Method	Venue	Importance score	Reserved	Training
StreamingLLM (Xiao et al., 2024)	ICLR 2024	position in the stream	newest M	–
H2O (Zhang et al., 2023)	NeurIPS 2023	attention accumulated over all queries, per head	newest $M/2$	–
Scissorhands (Liu et al., 2023)	NeurIPS 2023	share of the last 400 queries that give the token more than average attention, per head	newest 10	–
TOVA (Oren et al., 2024)	EMNLP 2024	attention of the last query averaged over heads, per layer	–	–
SnapKV (Li et al., 2024)	NeurIPS 2024	attention of the last w queries, average-pooled with kernel 5, per head	last w	–
PyramidInfer (Yang et al., 2024)	ACL Findings 2024	attention of the last queries weighted toward the newest, per layer, budget decreasing with depth	last $\min(32, M/8)$, newest 0.2 of the layer budget	–
DefensiveKV (Feng et al., 2026a)	ICLR 2026	SnapKV attention with a floor at its mean, weighted by the output norm of the value, budget shared across the heads of a layer	last w	–
LookaheadKV (Ahn et al., 2026)	ICLR 2026	attention of 32 learned lookahead queries, max-pooled with kernel 7, per head	tokens written after the probe	KL divergence to future attention, rank-8 LoRA active on the lookahead queries only, 500 steps
PyramidKV (Cai et al., 2025b)	COLM 2025	SnapKV score, budget decreasing linearly with depth	last 8	–
Ada-KV (Feng et al., 2025)	NeurIPS 2025	SnapKV score, budget shared across the heads of a layer	last w and $0.2M$ further per head	–
CAKE (Qin et al., 2025)	ICLR 2025	attention of the last w queries, layer budget from attention entropy and variance	last w	–
KeyDiff (Park et al., 2026)	NeurIPS 2026	negative cosine similarity to the mean key, per head	–	–
R-KV (Cai et al., 2025a)	NeurIPS 2025	0.1 attention minus 0.9 key redundancy, per head	last 8	–
ChunkKV (Liu et al., 2025)	NeurIPS 2025	SnapKV score averaged over chunks of 10 tokens, whole chunks kept, per head	last w	–
AudioKV (Wang et al., 2026c)	arXiv 2026	attention of the last queries low-pass filtered along the stream, head budget from a spectral prior	last 32 ($M/8$ for $M < 128$), half of M spread evenly	prior profiled on 32 Behavior-SD dialogues
VoxZip (Jia et al., 2026)	arXiv 2026	accumulated attention decayed by 0.95 every ten queries, per head	newest $0.2M$	–
DMS (Łańcucki et al., 2025)	NeurIPS 2025	eviction probability of the DMS adapter averaged over layers and heads	newest $\max(48, M/2)$, at most M	KL divergence and compression loss, window 48, 544 steps
TrimKV (Bui et al., 2026b)	ICLR 2026	learned retention decayed with the distance to the current step, per head	–	KL divergence to the Full KV cache, next-token and retention loss at memory 64, 2000 steps (MiniCPM-o 4.5), 1000 (Raon-SpeechChat)
Locret (Huang et al., 2024)	arXiv 2024	retaining head on queries, keys and values, per head	newest $0.42M$	Smooth L1 to the maximum future attention, 2000 steps
ForesightKV (Dong et al., 2026)	ICML 2026	MLP over keys, values and eight attention views, per head	newest $\min(64, M/2)$	pairwise ranking on future attention, 1000 steps

I Additional analysis

I.1 Cache composition, residency and removal

Under the Full KV cache, over the 200 sessions of FDB v1.0 user interruption, MiniCPM-o 4.5 writes 15.7 tokens per second and Raon-SpeechChat 27.6. On Raon-SpeechChat the agent writes an audio token at every frame whether or not the agent speaks, and 56.8% of these tokens are written while the agent is listening. Turn-control tokens, chunk markers and stop tokens together account for 23.4% on MiniCPM-o 4.5 because every step carries them, whereas the turn state accounts for 4.0% on Raon-SpeechChat. Under a policy that keeps the M most recent tokens, a token remains in the cache for M/r seconds, where r is the number of tokens written per second. At $M = 32$ this is 2.0 s on MiniCPM-o 4.5 and 1.2 s on Raon-SpeechChat, and the residency measured on recency-only rollouts is 1.99 s and 1.17 s. Covering the mean session of 28 s would require M of about 440 and 773. On Raon-SpeechChat, removing the user audio written while the user is silent also lowers the takeover rate by 0.79, whereas on MiniCPM-o 4.5 it does not change the judge score.

I.2 Cache budget and memory

Both backbones use a language model with 36 layers, 8 KV heads and a head dimension of 128 in bfloat16, so each cached token occupies $36 \times 8 \times 128 \times 2 \times 2$ bytes, or 144 KiB. At the token rates of Appendix I.1, the Full KV cache grows by 2.2 MiB per second on MiniCPM-o 4.5 and by 3.9 MiB per second on Raon-SpeechChat. A budget of M tokens per KV head holds the cache at $144\,M$ KiB in addition to the prefix and the tokens of the current step, whatever the length of the session. Table 9 compares the two and expresses each budget as the share of the Full KV cache that it retains.

Table 9: Memory of the Full KV cache and of a budget of M tokens per KV head, from the token rates of Appendix I.1 and 144 KiB per token. The share is M divided by the number of tokens written in 28 s (the mean FDB episode) or in 20 min.

	MiniCPM-o 4.5			Raon-SpeechChat		
	memory	28 s	20 min	memory	28 s	20 min
Full KV cache, 28 s	62 MiB	–	–	109 MiB	–	–
Full KV cache, 20 min	2.59 GiB	–	–	4.55 GiB	–	–
$M = 32$	4.5 MiB	7.3%	0.17%	4.5 MiB	4.1%	0.10%
$M = 64$	9.0 MiB	14.6%	0.34%	9.0 MiB	8.3%	0.19%
$M = 128$	18 MiB	29.1%	0.68%	18 MiB	16.6%	0.39%
$M = 256$	36 MiB	58.2%	1.36%	36 MiB	33.1%	0.77%

I.3 Accumulated attention and token distance

We record the attention of every query at layers 8, 17 and 26 under the Full KV cache, averaged over heads, and compute for each token its accumulated attention and its mean attention per query, the accumulated attention divided by the number of queries written after it. Table 10 gives the Spearman correlation of each with n_j per session, as the median over sessions, and Figure 15 shows layer 17. The accumulated attention is correlated positively with n_j in all 21 combinations of layer and type, and the mean attention per query negatively in 18.

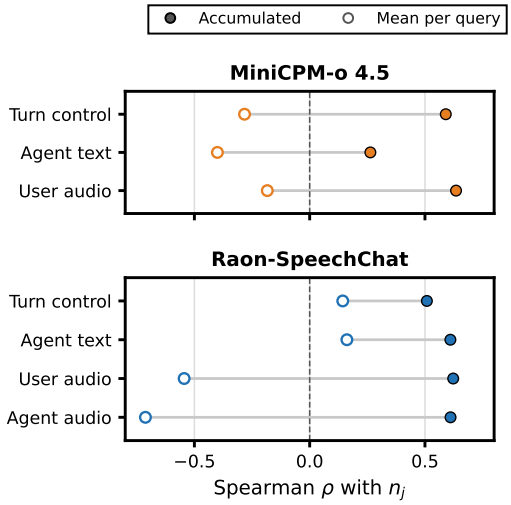

Figure 15: Correlation of the accumulated attention and of the mean attention per query with n_j under the Full KV cache on FDB v1.0 user interruption, layer 17.

I.4 Next-token loss and behavior

Table 11 lists every intervention of Section 3.3. The loss is measured on the reply generated with the Full KV cache, forced under each intervention, over all of its tokens and over its word tokens

Table 10: Median correlation of the accumulated attention (Acc.) and of the mean attention per query (Mean) with n_j under the Full KV cache on FDB v1.0 user interruption.

		Layer 8		Layer 17		Layer 26	
Backbone	Type	Acc.	Mean	Acc.	Mean	Acc.	Mean
MiniCPM-o 4.5	turn control	$+0.50$	-0.56	$+0.59$	-0.28	$+0.47$	-0.21
	agent text	$+0.29$	-0.57	$+0.26$	-0.40	$+0.36$	-0.47
	user audio	$+0.51$	-0.56	$+0.63$	-0.18	$+0.62$	-0.19
Raon-SpeechChat	turn control	$+0.56$	-0.28	$+0.51$	$+0.14$	$+0.50$	$+0.07$
	agent text	$+0.58$	-0.18	$+0.61$	$+0.16$	$+0.44$	-0.25
	user audio	$+0.69$	-0.61	$+0.62$	-0.54	$+0.27$	-0.61
	agent audio	$+0.45$	-0.75	$+0.61$	-0.71	$+0.93$	-0.73

only, which excludes padding, silence and turn-control tokens. On Raon-SpeechChat word tokens are 5.8% of the forced tokens and on MiniCPM-o 4.5 32.6%.

Table 11: Change in next-token loss and in behavior under each intervention on FDB v1.0 user interruption, relative to the Full KV cache with the same weights. Loss columns give the next-token loss over all tokens and over word tokens of the reply with the Full KV cache. A dash marks a setting in which no reply was rated.

Backbone	Intervention	Δ loss, all	Δ loss, words	Δ TOR	Δ judge score	empty
MiniCPM-o 4.5	remove chunk markers and stop token	$+0.09$	-0.10	-0.08	-1.36	0.08
	remove stop token	-0.12	-0.06	-0.08	-0.46	0.06
	remove turn control	$+3.93$	$+9.93$	-0.29	-4.03	0.30
	remove user audio	$+0.34$	$+1.37$	-0.32	-4.04	0.33
	remove agent text	$+2.66$	$+7.80$	-0.38	-3.96	0.40
	remove chunk markers	-0.57	-0.04	-0.09	-0.80	0.10
	remove user speech	$+0.48$	$+1.41$	-0.87	-3.86	0.90
	remove user silence	-1.38	$+0.08$	-0.08	$+0.01$	0.10
	no recency, $M = 32$	$+2.16$	$+4.23$	0.00	-3.83	0.00
	recency only, $M = 32$	$+0.44$	$+0.79$	0.00	-2.08	0.00
Raon-SpeechChat	remove agent audio	-0.15	-0.03	-0.86	-3.87	0.90
	remove user audio	$+0.91$	$+1.02$	-0.94	–	1.00
	remove agent audio while listening	-0.06	$+0.05$	-0.16	-0.05	0.21
	remove agent audio while speaking	-0.10	-0.10	-0.76	-3.44	0.77
	remove turn state	$+0.05$	$+2.15$	-0.89	-2.40	0.86
	remove agent text	$+0.35$	$+6.52$	-0.62	-3.69	0.58
	remove user speech	-0.25	$+0.75$	-0.47	-3.02	0.54
	remove user silence	-0.31	$+0.01$	-0.79	-1.86	0.82
	no recency, $M = 32$	$+0.60$	$+2.23$	-0.05	-3.78	0.10
	no recency, $M = 64$	$+0.67$	$+1.19$	$+0.05$	-3.71	0.01
	no recency, $M = 128$	$+0.66$	$+0.53$	$+0.02$	-3.62	0.03
	no recency, $M = 256$	$+0.52$	$+0.11$	$+0.02$	-2.59	0.03
	recency only, $M = 32$	-0.31	$+0.98$	$+0.03$	-2.67	0.03

I.5 Failure examples under a budget

Figure 16 shows replies from three settings of Section 3.3 in which the model takes the turn but the reply loses its content. Each example is an episode answered well with the Full KV cache, and the two replies follow the same interruption. Without the recency channel, both backbones produce fragments that do not address the request. With recency as the only channel at a small budget, Raon-SpeechChat replies fluently but to a different question.

I.6 Agreement between masked training and cache deletion

We measure the numerical agreement between the masked second pass and step-by-step decoding that deletes the same evicted keys, on ten agent channels from five Behavior-SD training windows per backbone at $M = 32$ and $M = 64$. The training mask is injected into decoding that removes the evicted keys from the cache. No key reappears after it is hidden, and every hiding falls on a decision boundary. With the output layer evaluated in float32, the two predict the same top-1 token at 98.9% or more of the positions on which the loss is taken, at $M = 32$ and $M = 64$ on

MiniCPM-o 4.5, inference weights 0.2:1:0 (no recency), $M = 32$

User interrupts: Actually, what are some financial goals I should set?

Full KV cache (score 5): You could aim to save for an emergency fund or maybe pay off some debt. Setting specific goals will help you stay motivated and focused on saving. Do you have any short-term goals in mind?

Under the budget (score 0): To rum um and show a like I

Raon-SpeechChat, inference weights 1:2:0 (no recency), $M = 64$

User interrupts: Oh, before I forget, how do I fix a leaky faucet?

Full KV cache (score 5): First, turn off the water supply, then check the washer inside the valve. If it's worn out, replace it. If not, you might need to replace the whole valve.

Under the budget (score 0): To decide us. You.

Raon-SpeechChat, recency only, $M = 32$

User interrupts: Oh, can we talk about something else? How about some travel tips?

Full KV cache (score 4): Sure thing. First off, always packet to usable water bottle to stay hydrated. Exactly. And try to avoid sugary drinks and snacks when you're on the row.

Under the budget (score 0): Sure thing. What kind of project are you working on?

Figure 16: Replies after the same interruption with the Full KV cache and under a budget on FDB v1.0 user interruption. Transcripts are the ASR output of the generated speech, reproduced verbatim.

both backbones, against 99.1% and 99.7% between an unmasked parallel pass and token-by-token decoding on MiniCPM-o 4.5 and Raon-SpeechChat.

I.7 Removing the gate channel

Figure 17 gives the judge score of DuplexKV with and without the gate channel.

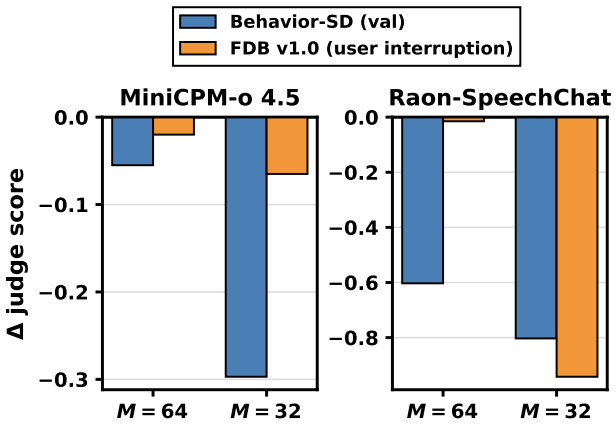

Figure 17: Effect of removing the gate channel ($w_g = 0$ in Eq. (10)) on the judge score of DuplexKV.