MemoLens: Empowering Augmented Reality Glasses with Super Memory

Samiul Alam, Shakhrul Iman Siam, Mi Zhang

The Ohio State University
Columbus, Ohio, USA
{alam.140,siam.5,mizhang.1}@osu.edu

Abstract

Spatial computing empowered by augmented reality (AR) glasses emerges as a new computing paradigm. One of its most transformative capabilities is *super memory* – the super power to accurately recall objects and people an individual has been paying attention to or interacting with in the physical world. In this work, we propose MemoLensthat empowers AR glasses with super memory. Achieving such capability is not trivial: it requires storing vast amounts of visual data as compact digital memory and retrieving relevant information from the memory efficiently when being prompted. MemoLens achieves this via two key innovations. First, MemoLens leverages eye gaze information captured by the AR glasses to detect what the individual is paying attention to, and introduces a spatio-temporal token compression technique to generate highly compact visual memory. Second, MemoLens introduces a hierarchical search scheme that enables efficient retrieval of relevant memory based on user prompts. We implemented MemoLens using Meta's Aria AR glasses, and evaluated its performance on more than 100 hours of egocentric videos collected in real-world settings. Our results show that MemoLens is able to achieve accurate memory retrieval in real time. Given its promising performance, we believe MemoLens represents a significant step towards realizing always-on super memory in next-generation AR glasses. The project's homepage is https://aiot-mlsys-lab.github.io/memolens.github.io/.

CCS Concepts

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; **Mobile computing**; • **Information systems** → **Information retrieval**; *Multimedia information systems*; • **Computing methodologies** → *Machine learning algorithms*; **Computer vision tasks**.

Keywords

Augmented Reality, Egocentric Vision, Eye Gaze, Memory Augmentation, Visual Retrieval

ACM Reference Format:
Samiul Alam, Shakhrul Iman Siam, Mi Zhang. 2026. MemoLens: Empowering Augmented Reality Glasses with Super Memory. In *The 24th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '26), June 21–25, 2026, Cambridge, United Kingdom.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3745756.3809235

MobiSys '26, Cambridge, United Kingdom

ACM ISBN 979-8-4007-2027-7/2026/06
https://doi.org/10.1145/3745756.3809235

1 Introduction

The computing landscape is undergoing a paradigm shift from handheld smartphones to head-worn augmented reality (AR) glasses [3, 32, 47]. Modern AR glasses are increasingly equipped with sophisticated multimodal sensor arrays, including always-on egocentric cameras, microphone arrays, and Inertial Measurement Units (IMUs) [10, 16, 36]. This has given rise to spatial computing that enables users to interact with the physical world in new ways with AR devices.

Crucially, state-of-the-art AR glasses have now integrated eye-tracking, allowing for more niche applications in attention monitoring [4, 13, 22]. This convergence of capability between continuous first-person video acquisition and precise attentional monitoring creates an unprecedented opportunity: the ability to build intelligent user agents that do not merely react to explicit commands, but actively observe, interpret, and remember the user's important daily experiences [37, 48].

A transformational application of this capability is *super memory* – the super power to accurately recall objects and people an individual has been paying attention to or interacting with in the physical world. Although conceptually similar to "lifelogging" [15, 17], super memory differs fundamentally in its data philosophy. Traditional lifelogging systems operated on the principle of *indiscriminate capture*, archiving vast amounts of redundant data that could overwhelm both storage subsystems and the user's ability to efficiently retrieve information [24, 39]. We argue that an effective memory augmentation system must be *selective*, mimicking biological memory by prioritizing information that enters the user's focus.

In this work, we introduce MemoLens, a spatial computing framework that empowers AR glasses with super memory.

Figure 1 illustrates an envisioned scenario of MemoLens. As shown in Figure 1 (a), MemoLens keeps track of the objects the user attends to throughout the day. When he realizes he has forgotten where he put his keys, as shown in Figure 1 (b), he asks MemoLens "Can you recall where I left my keys?" MemoLens processes the query and generates a precise, context-aware answer: "They are on the desk in your reading room."

Realizing this vision is not trivial and requires MemoLens to address two key challenges.

Challenge#1: Creating Compact and Informative Memory. Continuous recording of high-resolution egocentric videos generates a gigantic volume of data. Simply storing raw videos as memory is impractical, as semantic information cannot be instantly

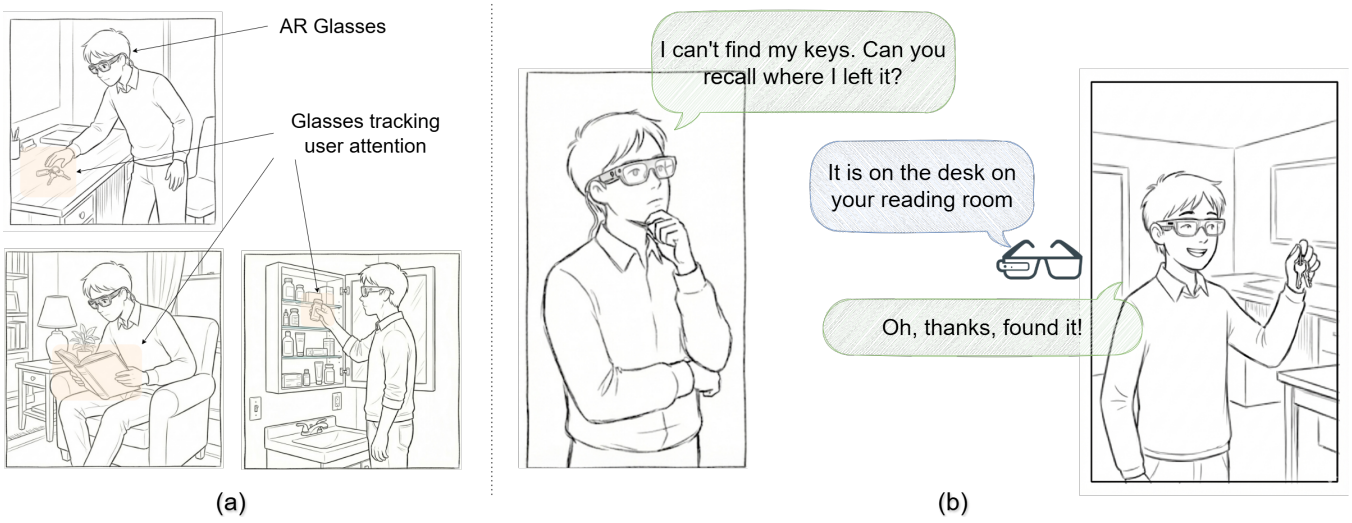


Figure 1: Illustration of an envisioned scenario of MemoLens.

extracted for practical interaction. As such, a fundamental challenge lies in distilling this large volume of raw video into a compact and semantically informative memory, as human memory is.

Challenge#2: Efficient Memory Retrieval. For a memory assistant to be practically useful, memory retrieval needs to be near-instantaneous. However, searching through a history of days or even weeks of memory involves computing similarity scores across millions of memory records. Therefore, the challenge lies in designing a scheme that enables MemoLens to search through the created memory and retrieve the correct memory record in real time.

To address the first challenge, the key design principle behind MemoLens is to leverage *eye gaze* as a proxy of user attention to retain visual information within video frames that a user actually attends to [26, 33]. Such attended information serves as the foundation of memory formation [2, 20]. Extensive research in cognitive science and egocentric vision has established a strong link between eye gaze, human activity, and memory retention, demonstrating that users consistently fixate on objects crucial to their current tasks and future recall [26, 33, 44].

Building upon this principle, MemoLens partitions each incoming video frame from the AR glasses into patches, identifies patches that are important based on eye gaze, and extracts information from those important patches to generate a compact representation for each frame. As a result, MemoLens creates compact and semantically informative memory for each video frame, which we refer to as *memory snippet*, that can be directly used for retrieval and text generation in the next stage.

To address the second challenge, rather than searching through all the generated memory snippets one by one, MemoLens first arranges them into a hierarchical tree structure in which leaf nodes encode detailed short-term memory, while higher-level nodes summarize higher-level, longer-term memory. When given a query, MemoLens adopts a coarse-to-fine top-down retrieval mechanism to traverse down the tree from the top to retrieve the most relevant memory snippets in an efficient manner.

System Implementation & Experimental Results. We implemented MemoLens as a distributed framework that spans between a pair of AR glasses, a smartphone, and a cloud server. We have conducted a comprehensive set of experiments to evaluate the performance of MemoLens in three aspects: 1) retrieval accuracy, 2) effectiveness of eye gaze, 3) generation consistency, and 4) system performance in terms of latency, storage, and energy consumption.

Our results show that

- **Retrieval Accuracy:** MemoLens maintains almost baseline performance at moderate compression for two different backbone models. For X-Clip, it maintains over 96% top-3 accuracy at 47% token reduction, while CLIP4Clip sustains this high accuracy up to 62.5% reduction. Even under aggressive compression (94%), both models retain over 78% accuracy. Notably, at this level, MemoLens significantly outperforms standard baselines: X-Clip surpasses cross-frame merging by over 26%, while CLIP4Clip maintains performance where baselines collapse to below 50% accuracy.
- **Effectiveness of Eye Gaze:** Our ablation studies confirm that eye gaze is critical for efficient memory creation, especially at high reduction ratios (94%). Gaze-guided merging boosts X-Clip retrieval accuracy by approximately 14% (from 64.1% to 78.4%) and 17% for CLIP4Clip (from 60.9% to 78.2%) compared to merging without gaze constraints at that ratio.
- **Generation Consistency:** The generated natural language responses maintain a BERTScore of over 88% relative to uncompressed video, validating that semantic context is preserved. The generated responses maintain close similarity up to 60% token reduction ratio for LLaVa backbone and 40% reduction ratio for Llama backbone.
- **System Performance:** MemoLens achieves sub-linear retrieval latency via its hierarchical tree structure, ensuring scalability for long-term memory. As a highlight, MemoLens can achieve real-time responsiveness with a median Time-To-First-Token (TTFT) of 69ms for response generation.

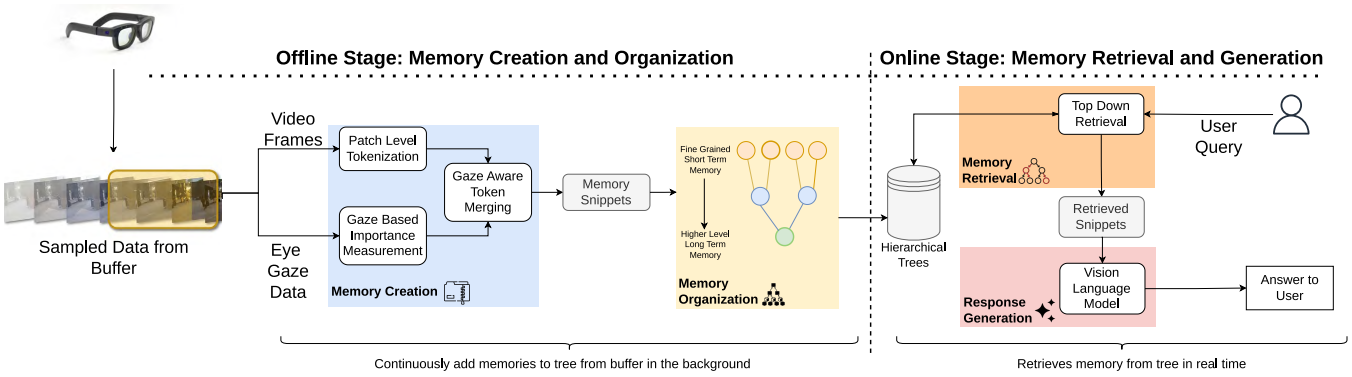


Figure 2: Overall architecture of MemoLens.

Summary of Contributions. To the best of our knowledge, MemoLens represents the first spatial computing framework that empowers AR glasses with super memory. It contributes key techniques that address the unique challenges in AR glasses-powered super memory. We believe our work represents a significant step towards turning the envisioned super memory into reality.

2 MemoLens Overview

Figure 2 illustrates the overall architecture of MemoLens. As shown, MemoLens involves two stages: an offline *memory creation & organization* stage and an online *memory retrieval & generation* stage. We explicitly note that "Offline" in this context refers to background indexing; MemoLens processes the continuous video streams in near real-time using a sliding window buffer to update the memory tree, rather than processing after the fact.

The offline stage consists of two phases: memory creation (§3.1), and memory organization (§3.2).

In the memory creation phase, MemoLens first converts each video frame streamed from AR glasses into tokens at the patch level (*Patch Level Tokenization*). Based on the measured importance of each patch using eye gaze information (*Gaze-based Importance Measurement*), it then extracts information from important tokens to generate a compact representation at the frame level (*Gaze-aware Token Merging*). These compact representations act as the building blocks of the super memory, which we refer to as *memory snippets*.

In the memory organization phase, MemoLens organizes the memory snippets hierarchically in the form of trees, where lower-level nodes store fine-grained short-term memory, while higher-level nodes capture higher-level memory over longer temporal periods.

Finally, in the online memory retrieval & generation stage (§3.3), given a user query, MemoLens performs top-down retrieval throughout the tree to retrieve the corresponding memory snippets, and pass them to a vision–language model to generate an answer back to the user.

3 Design of MemoLens

3.1 Memory Snippets Creation

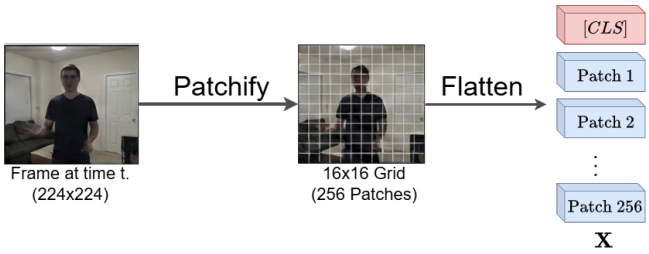


Figure 3: Illustration of patch level tokenization.

3.1.1 Patch Level Tokenization. As the very first step, video frames are uniformly sampled at one frame per second and resized to 224 by 224 pixels to match the model's input format. As shown in Figure 3, each sampled frame is divided into a 16×16 grid, yielding $P = 256$ non-overlapping patches. These patches undergo flattening and linear projection. Following ViT [11], a learnable token [CLS] is prepended to aggregate global information, resulting in the sequence of tokens $\mathbf{X}$ as:

$$\mathbf{X} = \{[\text{CLS}], \text{Patch}_1, \ldots, \text{Patch}_P\}, \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{B \times N \times D}$, B is the temporal dimension, N is the sequence length, and D is the feature dimension.

3.1.2 Gaze-based Importance Measurement. Given its high-frequency movements, eye gaze is sampled at 10 Hz. Therefore, ten eye gaze points are collected for each video frame. Given the sampled eye gaze points, our objective is to use them to measure the importance of every patch within each video frame. To achieve this, as shown in Figure 4, the eye gaze points are first projected onto the frame's pixel coordinates. To spread the influence of a single gaze point across a wider area within the frame, a convolution operation with a Gaussian kernel is applied to each gaze point to generate a continuous pixel-level heatmap. This heatmap is also divided by the 16×16 grid to generate 256 non-overlapping patches. Finally, we apply average pooling within each patch region, followed by normalization to derive an importance score. This discretizes the continuous heatmap into a grid of scalar importance scores that map 1-to-1 with the visual patches. We then flatten the grid. This results in a vector, p, consisting of 256 importance weights.

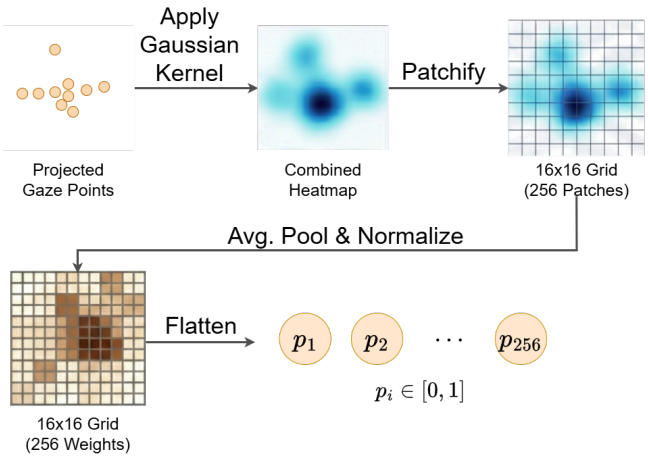


Figure 4: Illustration of gaze–based importance measurement.

3.1.3 Gaze-aware Token Merging. In this stage, we use the previously created token sequence $\mathbf{X}$ and the gaze importance weight, p. For retrieval and generation tasks, the sequence $\mathbf{X}$ is passed through a vision transformer to create a representation for each patch that can be projected to the vector space of text tokens. We utilize gaze-aware token merging in the vision transformer to reduce the length of the output sequence. To do this, we insert *Gaze Aware Token Merging* layers between each of the transformer blocks in the vision transformer model. We adopt merging rather than pruning, enabling MemoLens to retain background information that may be useful. The goal of the token merging layer is to reduce N gradually in each layer, resulting in a smaller output sequence length and memory savings. Without the token merging layer, the output sequence would also have N tokens. Each layer reduces the sequence length by δ tokens per frame (or per temporal pair) while preserving critical information. The algorithm is shown in Figure 5. The output generated is a more compact token sequence, $\mathcal{H}^{(0)}$, which we define as *memory snippets*. Conceptually, each merging layer operates in four stages:

(1) Source/target partition. Let $U = \{1, \ldots, N\}$ be the set of token indices for a given frame. We utilize gaze-based importance scores p to partition tokens into a source set $\mathcal{S}$ representing tokens that can be merged, and a target set $\mathcal{T}$, or destinations onto which the tokens will be merged. We first determine the source pool capacity S, as defined in (2), based on the ratio parameter k and the reduction target δ, ensuring the pool is large enough to support δ merges if possible:

$$S = \max\left(\delta, \left\lfloor \frac{k}{1+k} N \right\rfloor\right). \tag{2}$$

We set a protected set $Q \subset U$ (e.g., the [CLS] token) that is forced to be a target and the similarity scores are always set to negative infinity, essentially blocking them from merging. From the unprotected tokens, we select the S tokens with the lowest importance scores p, as shown in Eq. 3:

$$\mathcal{S} = \underset{A \subseteq (U \setminus Q),\, |A| = S}{\arg\min} \sum_{i \in A} p_i, \quad \mathcal{T} = U \setminus \mathcal{S}. \tag{3}$$

(2) Candidate scoring and strategy selection. MemoLens dynamically selects between two merging strategies: *Spatial* (intra-frame) or *Temporal* (inter-frame). Tokens are projected to key vectors $\mathbf{K}$ and normalized to $\hat{\mathbf{K}}$. To decide the strategy, we first compute a similarity score V for the batch by summing the cosine similarities of the top-δ_{eff} matches, where $\delta_{\text{eff}} = \min(\delta, S)$.

For **Spatial** scoring, matches are restricted to the same frame t. We calculate the score using Eq. 4:

$$V_{\text{spatial}} = \sum_{t=0}^{B-1} \sum_{i \in \text{top-}\delta_{\text{eff}}(\mathcal{S}_t)} \max_{j \in \mathcal{T}_t} (\hat{\mathbf{k}}_i^\top \hat{\mathbf{k}}_j);\ \hat{\mathbf{k}} \in \mathbb{R}^D. \tag{4}$$

For **Temporal** scoring, we treat the batch as a sequence of time steps and consider disjoint pairs of frames $(2t, 2t+1)$. The target pool for a source includes targets from both its own frame and its paired frame. We calculate the temporal score using Eq. 5:

$$V_{\text{temporal}} = \sum_{t}^{B/2-1} \sum_{i \in \text{top-}\delta_{\text{eff}}(\mathcal{S}_{2t} \cup \mathcal{S}_{2t+1})} \max_{j \in \mathcal{T}_{2t} \cup \mathcal{T}_{2t+1}} (\hat{\mathbf{k}}_i^\top \hat{\mathbf{k}}_j);\ \hat{\mathbf{k}} \in \mathbb{R}^D. \tag{5}$$

If the temporal score exceeds the spatial score, we execute temporal merging; otherwise, spatial merging is performed.. We can bias the model toward either by adding a decision margin γ.

(3) Best-match assignment. Given the selected strategy, we determine the specific assignments. For every source token $i \in \mathcal{S}$, we search for the best match j^* within the valid target set $\mathcal{T}_{\text{valid}}$ defined by the strategy shown in Eq. 6:

$$j^*(i) = \underset{j \in \mathcal{T}_{\text{valid}}}{\arg\max} (\hat{\mathbf{k}}_i^\top \hat{\mathbf{k}}_j). \tag{6}$$

- **Spatial:** $\mathcal{T}_{\text{valid}} = \mathcal{T}_t$ (intra-frame only).
- **Temporal:** For a pair of frames (t_1, t_2), a source $i \in \mathcal{S}_{t_1} \cup \mathcal{S}_{t_2}$ searches targets in $\mathcal{T}_{\text{valid}} = \mathcal{T}_{t_1} \cup \mathcal{T}_{t_2}$.

We rank all pairs $(i, j^*(i))$ globally by similarity and retain only the top δ_{eff} pairs. Sources not selected are designated as "unmerged tokens" and are preserved in the output sequence.

(4) Merge update. We merge sources into their assigned targets using a weighted average. Let s_i denote the size (number of raw patches aggregated) of token i. For a target j assigned a set of sources $\Omega_j = \{i \mid j^*(i) = j\}$, we update its feature vector $\mathbf{x}_j$, size s_j, and importance score p_j according to Eq. 7:

$$\mathbf{x}'_j = \frac{s_j \mathbf{x}_j + \sum_{i \in \Omega_j} s_i \mathbf{x}_i}{s'_j}, \tag{7}$$

$$p'_j = \frac{s_j p_j + \sum_{i \in \Omega_j} s_i p_i}{s'_j}, \tag{8}$$

$$s'_j = s_j + \sum_{i \in \Omega_j} s_i. \tag{9}$$

The merged sources are subsequently removed from the sequence.

After merging, each layer reduces the length of the sequence by δ. So for $\mathcal{L}$ transformer blocks, the number of tokens in the output sequence will be $(N - \mathcal{L}\delta)$. $\mathcal{L}\delta/N$ is the token reduction ratio, r.

After merging $\mathcal{L}$ layers, we get memory snippets, $\mathcal{H}^{(0)}$. An extra token called the *summary token* is prepended to the sequence, representing the entire snippet. This is computed differently for different models, e.g., for CLIP4Clip, this is the mean of all [CLS]

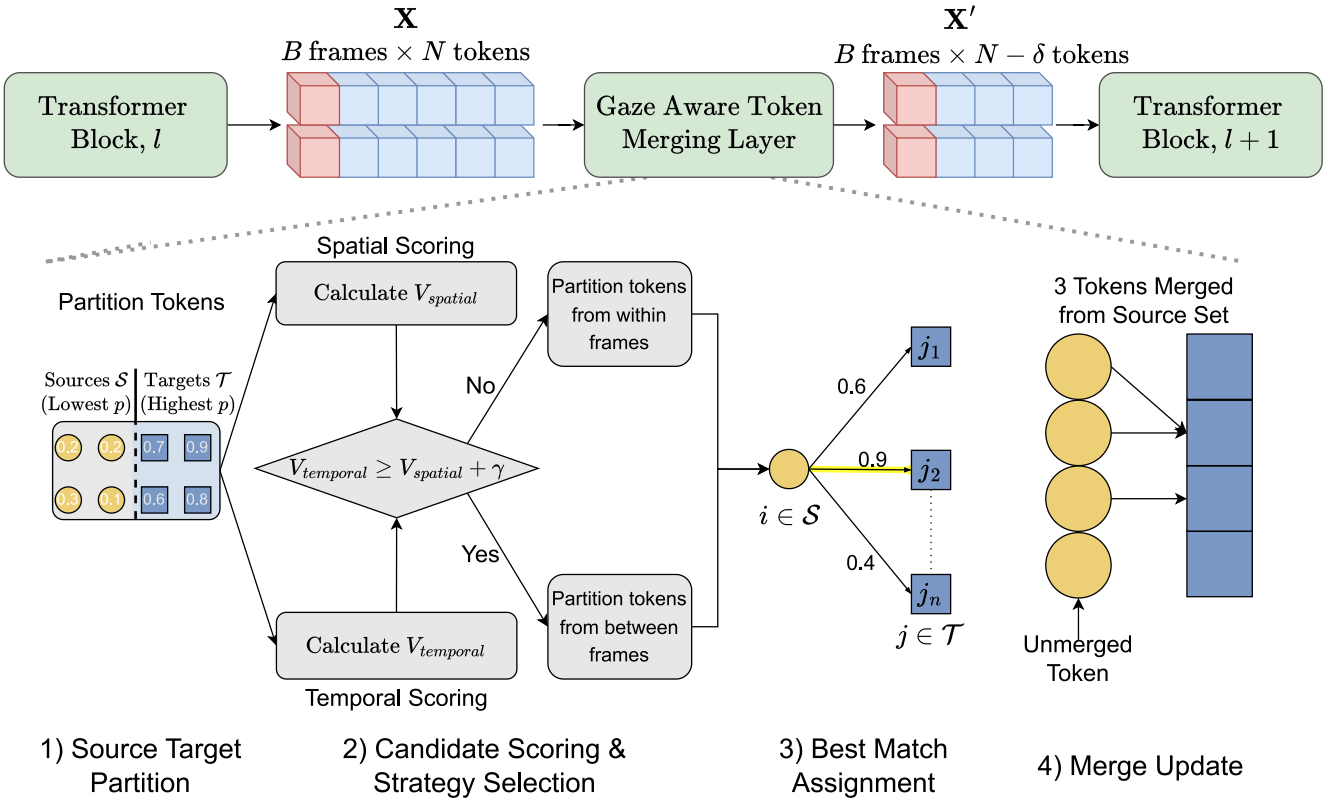


Figure 5: Illustration of gaze-aware token merging.

tokens, while for X-Clip, it is computed using a separate module. This is passed to the next stage §3.2 as Level 0 tokens.

3.2 Memory Organization

In this stage, we use the memory snippets $\mathcal{H}$ and store them in a way that allows for fast retrieval. To efficiently index long-term egocentric data, we structure the memory snippets into a multi-level tree of increasing temporal abstraction. The leaves of this tree, denoted as **Level 0**, are constructed directly from the Memory Creation phase. Specifically, we extract the *summary token* from each memory snippet to serve as the atomic unit of the hierarchy. Let $\mathcal{H}^{(0)} = \{\mathbf{h}_1^{(0)}, \mathbf{h}_2^{(0)}, \ldots, \mathbf{h}_M^{(0)}\}$ represent the sequence of Level 0 embeddings, where $\mathbf{h}_i^{(0)}$ corresponds to the `[CLS]` embedding of the i-th snippet.

To generate higher-level nodes, we employ a recursive aggregation function, denoted as $f(\cdot)$. As illustrated in Figure 6, this function is instantiated as a 4-layer Transformer Encoder we define as the *cross-frame integration module*. Each layer consists of a Multi-Head Attention (MHA) module with 8 heads and a Feed-Forward Network (FFN) with a hidden size of 512 and an intermediate size of 2048, with residual connections and layer normalization applied after each block. The architecture follows [31]. We apply a Mean Pooling layer at the final stage to generate a summary of the input temporal window. Mathematically, the hierarchy is constructed recursively. For a given level $l > 0$, the sequence of embeddings

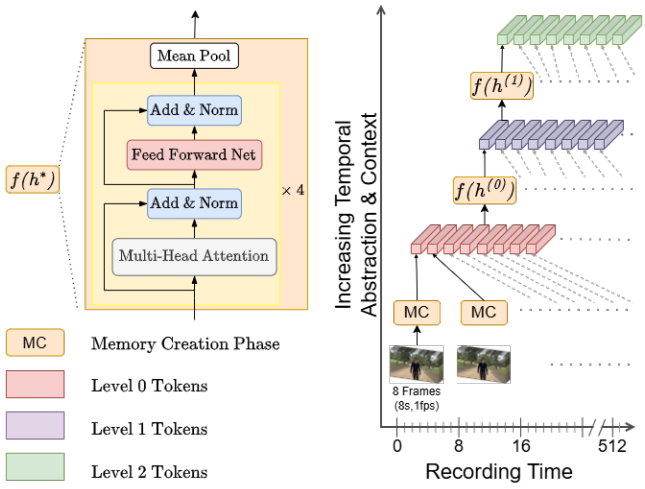


Figure 6: Illustration of hierarchical memory organization.

$\mathcal{H}^{(l)}$ is derived by sliding a non-overlapping window of size $W = 8$ over the sequence from the previous level $\mathcal{H}^{(l-1)}$. The embedding for the j-th node at level l is computed as:

$$\mathbf{h}_j^{(l)} = f\left(\left[\mathbf{h}_{j\cdot W}^{(l-1)}, \ldots, \mathbf{h}_{(j+1)\cdot W-1}^{(l-1)}\right]\right) \tag{10}$$

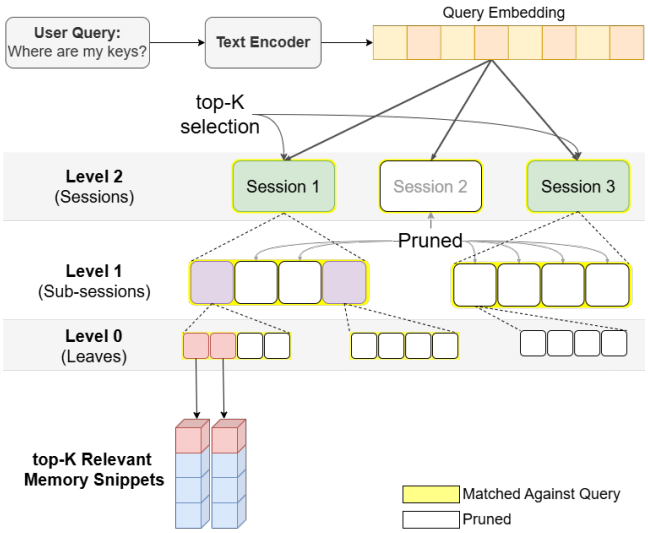


Figure 7: Illustration of memory search.

We choose depth $L = 3$ and window length $W = 8$ as these hyper-parameters yielded the best empirical performance. Level 1 tokens are created by aggregating 8 consecutive Level 0 tokens (approx. 64 seconds). Likewise, level 2 tokens are created by aggregating 8 Level 1 tokens (approx. 512 seconds) into a high-level activity node. This results in a tree of depth 3 where the root covers long-term contexts and edges represent finer detail.

3.3 Memory Retrieval & Response Generation

3.3.1 Memory Retrieval. Once the memory trees are constructed, MemoLens can efficiently respond to user queries through a coarse-to-fine *top-down retrieval* mechanism.

Let the hierarchical structure be defined with levels $l \in \{0, 1, 2\}$, where Level 2 is the root and Level 0 are the leaves. As defined in §3.2, $\mathcal{H}^{(l)}$ is the set of all node embeddings at level l. For a node embedding $\mathbf{h}$ at level $l > 0$, let $Ch(\mathbf{h}) \subset \mathcal{H}^{(l-1)}$ represent the set of its children embeddings. Let $\text{Select}(\mathcal{H}, K)$ be an operator that takes a set of memory embeddings $\mathcal{H}$ and returns the subset of size K with the highest cosine similarity to the query $\mathbf{q}$. Formally:

$$\text{Select}(\mathcal{H}, K) = \underset{\mathbf{h} \in \mathcal{H}, K}{\arg \text{topK}} \, \text{sim}(\mathbf{q}, \mathbf{h}). \tag{11}$$

The retrieval process is defined by determining the *Candidate Set* $C^{(l)}$ and the *Selected Set* $\mathcal{M}^{(l)}$ at each level l.

For any level, l, the *candidate set*, $C^{(l)}$ is constructed by concatenating only the children of the nodes selected in the previous level $(l + 1)$. We select the top-K_l nodes from this restricted pool to get the *selected set*:

$$C^{(l)} = \bigcup_{\mathbf{h} \in \mathcal{M}^{(l+1)}} Ch(\mathbf{h}), \tag{12}$$

$$\mathcal{M}^{(l)} = \text{Select}(C^{(l)}, K_l). \tag{13}$$

This process is illustrated in Figure 7. As noted in §3.2, the depth of the tree used in this work is three. We use $K = 3$, as we are evaluating the retrieval model using top-3 accuracy. At the root level, the search space includes all available level 2 tokens, or *sessions*. So, the initial candidate set and select set will be defined as follows:

$$C^{(2)} = \mathcal{H}^{(2)}, \tag{14}$$

$$\mathcal{M}^{(2)} = \text{Select}(C^{(2)}, K_2). \tag{15}$$

We descend into subsequent levels using (12) to first obtain the level 1 tokens or *sub-sessions* and finally get the *leaf* nodes or Level 0 tokens. Based on the selected *leaves*, we take the corresponding memory snippets, which are the entire token sequence for that leaf. Because we have pruned most of the tree, the total number of similarity evaluations scales sub-linearly with the number of stored frames, improving latency. The selected memory snippets, along with the user query embedding $\mathbf{q}$, are then sent to the next stage for response generation.

3.3.2 Response Generation. Following the retrieval process, MemoLens proceeds to synthesize a natural language response. We arrange the selected set of memory snippets from the retrieval stage chronologically to preserve temporal order. To generate the final answer, we employ a multi-modal large language model for response generation. We construct a multi-modal prompt that conditions the model on the memory snippets and the user's original query. Specifically, the retrieved visual sequences are embedded as context. Because these visual embeddings may reside in a different feature space than the language model's expected inputs, we utilize a trained adapter module to project the aggregated visual embeddings directly into the Large Language Model's (LLM) input space, ensuring proper vision-language alignment. This is followed by an instruction to answer the query based strictly on the provided context. The model then processes this multi-modal context auto-regressively to generate a natural language answer. This response is finally presented to the user via the AR interface.

4 System Implementation

We implemented MemoLens as a distributed framework that spans across a pair of Meta Aria AR glasses, a Samsung S23 Ultra smartphone, and a cloud server with an NVIDIA A100 GPU. The egocentric video and eye gaze data captured by the Meta Aria AR glasses are first transmitted to the smartphone. The video and gaze data are transformed as discussed in §3.1.1 and §3.1.2. This is done using native Kotlin. The outputs of this step are then passed to the gaze-aware token merging module. The tokens are then transformed to output tokens with a Vision Transformer. We insert *Gaze Aware Token Merging* layers between each of the transformer blocks in the vision transformer model. The customized model is implemented in PyTorch for inference in cloud and also exported for the Android ecosystem using the Onnx library with a fixed token reduction ratio. The user has the choice of either processing in the cloud or on the phone. The outputs in this step are the compact memory snippets. From this, we further create the hierarchical memory tree using the cross-frame integration module defined in §3.2. This is also implemented in PyTorch and exported to Android with Onnx. When the glasses receive an audio prompt, it is transcribed to text on the device and sent to the phone. The phone performs a similarity search over the hierarchical tree §3.3.1 to select the most relevant embeddings. This is implemented in Kotlin. The selected

embeddings are sent to the cloud, which projects these embeddings into the LLM's input space via a trained adapter layer and generates text. The response is conveyed to the user via the glasses' or smartphone's audio interface.

5 Evaluation

5.1 Experimental Methodology

5.1.1 Datasets. Most existing egocentric datasets lack the scale or diversity of eye gaze and video data required for effective finetuning of MemoLens. For example, in Ego4D [16], which is a standard benchmark, the subset containing gaze data is limited to ∼29 hours. We therefore utilize the Nymeria dataset [30] collected using the Meta Aria Glasses and curate a 100-hour subset containing RGB video (1408 × 1408, 30 fps), synchronized eye-tracking (10 fps), and detailed annotations. The annotations consist of fine-grained *atomic narrations* (spanning ∼ 5s) and high-level *activity summaries* (spanning ∼ 30s). Since the captions in the dataset are not annotated in uniform segments, we generate ground-truth captions for the video segments using the Qwen-2.5-VL model at 8s intervals. The model is prompted to generate captions from video segment and the original human annotations available for a segment.

5.1.2 Models and Finetuning. We utilize X-Clip [31] and CLIP4Clip [28] as backbones and fine-tune them on the Nymeria dataset using a two-stage training procedure. In the first stage, the base models were fine-tuned on the Nymeria dataset. We train only the top 3 layers of the vision encoder and the cross-frame integration head of this model, and keep all the other layers frozen. We use a batch size of 128 and use contrastive loss to finetune the model. The model was finetuned for 5 epochs with a learning rate of 10^{-4}. In the second stage, we fine-tune the cross-frame integration module. The module takes 8 frame embeddings and aggregates them into a single embedding representing the snippet. This was done recursively for each level We compute the embeddings and construct text-embedding pairs for each level. The transformer heads were fine-tuned for 15 epochs with a learning rate of 10^{-4} with early stopping. We use contrastive loss and maximize the similarity of a node with all its descendants while minimizing similarity with non-descendants at lower levels. All training is conducted using an AdamW optimizer.

5.1.3 Baselines and Evaluation Metrics. We compare MemoLens against several baselines across three dimensions: retrieval accuracy, generation consistency and search efficiency. Below, we outline the baselines considered and the metrics used to evaluate performance.

Memory Retrieval Accuracy. We compare against standard X-Clip and CLIP4Clip architectures with **No Token Merging** and measure how much accuracy is preserved at different token reduction ratios. Additionally, we compare against **ToMe** [5] based strategies: (a) *Intra-frame merging* (spatial only) and (b) *Cross-frame merging* (temporal only) to evaluate the spatio-temporal merging mechanism in MemoLens. In the intra-frame case, all transformer layers perform token merging only within the same frame. Likewise, in cross-frame token merging, all transformer layers perform token merging only between consecutive frames. We quantify retrieval performance using *Top-3 accuracy*. Due to the large scale of the dataset, we adopt batched evaluation. The test set is partitioned into M randomized subsets, where each subset contains 64 pairs of video clips (8s duration) and their corresponding text annotations. For a given query, the model ranks all B clips in the subset based on cosine similarity. We define the accuracy as the percentage of queries where the ground-truth video clip appears in the top-3 ranked results, averaged across all M subsets.

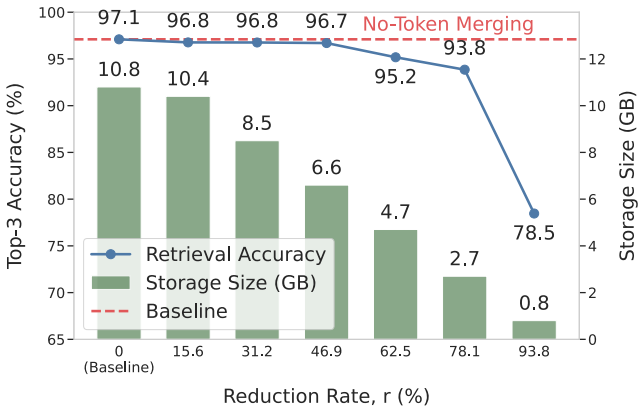
Generation Consistency. We benchmark against LLaVa-NeXT-Video [51] and Llama-3.2-11B-Vision-Instruct using full video embeddings (no compression). Video is added to the input as a list of images. Our objective is to verify if the computed memory snippets hold the critical information under high token merging. To measure the information retention, we utilize the BERTScore F1 metric [50]. We treat the text generated by the VLM using the full, uncompressed video embeddings as the reference text. We compare this against the candidate text generated by the same VLM when conditioned on our memory snippets. The BERTScore computes the semantic similarity between the candidate and reference text via cosine similarity of BERT embeddings.

Search Efficiency. We compare our hierarchical tree search against a *Linear Search* where the query is compared against every token individually. We measure the inference time for searching when the size of the datastore to search is varied from 1 minute to 4 hours, corresponding to the data store containing 1 clip and 4000 video clips, respectively.

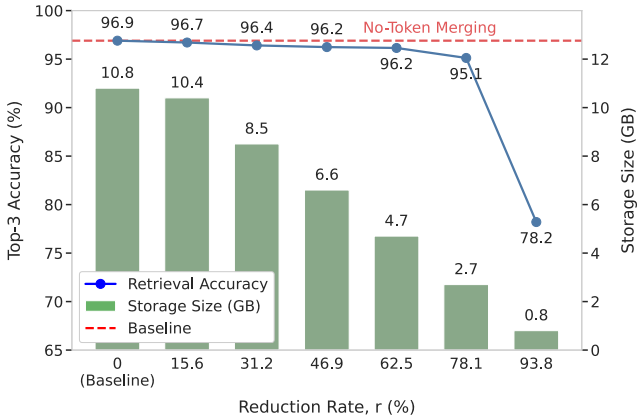
5.2 Memory Retrieval Accuracy

5.2.1 Accuracy under Different Token Reduction Ratios. We begin as mentioned in §5.1.3, by comparing the overall retrieval performance of MemoLens across two base models, X-Clip and CLIP4Clip, at different token reduction ratios against no token merging. The results are shown in Figure 8. The token reduction ratio, r represents the percentage of tokens merged. The total storage footprint of a 2-hour video is shown as well for reference. Across both base models, MemoLens maintains near-baseline accuracy at low to moderate reduction levels. For X-Clip, MemoLens achieves retrieval accuracy above 96% for token reduction ratios at or below 46.9%, and remains above 93% through 46.9% and 62.5% reduction. CLIP4Clip shows a similar pattern, with accuracy staying above 96% up to 62.5% and above 95% at 78.1%. At the highest token reduction ratio of 93.8%, retrieval accuracy drops more noticeably but still remains above 78% for both models, despite the memory footprint shrinking more than 13 times to under 1 GB.

5.2.2 Superiority of Spatio-Temporal Token Merging. As discussed in §5.1.3, we compare MemoLens against the 2 baselines: Intra-frame Token Merging and Cross-frame Token Merging. The top-3 retrieval accuracy on both X-Clip and CLIP4Clip, are shown in Figure 9 (a). As shown, for both backbone models, MemoLens outperforms the token merging baselines at all token reduction ratios. For X-Clip, we see a maximum top-3 retrieval accuracy of 91.79% at 15.625% token reduction compared to 89.45% for cross-frame merging and 89.38% for intra-frame merging. In this case intra-frame and cross-frame merging performs almost exactly same. The retrieval accuracy using the full token sequence is 93.3%. Moreover, MemoLens retains accuracy even under high token compression,



(a) X-Clip



(b) CLIP4Clip

Figure 8: Retrieval performance of MemoLens vs no-token merging.

dropping only 8%. The gap between MemoLens and other token merging baselines widens when the ratio of merged tokens increases as well. At 93.75% reduction, the accuracy is 84.07%, which is 29.17% higher than intra-frame merging and 26.27% higher than cross-frame merging. For CLIP4Clip, up to 60% reduction, the three curves almost overlap, showing that all methods preserve retrieval accuracy well. At 90% reduction, however, MemoLens still achieves near 80% accuracy, while cross-frame merging drops to the low 60% range and intra-frame merging collapses to below 50%.

5.2.3 Benefits of Leveraging Eye Gaze Modality. To test how well gaze correlates to the user's attention and focus, we compare the performance of regular token merging with the gaze-weighted merging scheme used in MemoLens. We set up the experiment as discussed in §5.1.3, we compare the performance of MemoLens using both X-Clip and CLIP4Clip models with and without taking gaze weights into consideration. Figure 9 (b) shows the top-3 retrieval accuracy of token merging with and without considering eye gaze for X-Clip and CLIP4Clip. Both figures follow a similar pattern. As shown, the performance is nearly identical at low token reduction ratios. However, for high rates of token reduction, the retrieval accuracy for token merging without eye-gaze falls off faster when the reduction ratio is increased. For X-Clip, at a 94% token reduction ratio, retrieval accuracy drops to 64.1% without gaze compared to 78.4% with gaze. Likewise, for CLIP4Clip, accuracy at the same token reduction ratio is 60.9% without gaze versus 78.2% with gaze. For both models, the curve for token merging using eye gaze data declines more gradually, mirroring the main results in Figure 9 (b). This indicates that eye gaze acts as a good filter, especially under a high token reduction ratio, by selectively emphasizing the tokens most likely to contribute to accurate retrieval. Appendix 10 discusses the tradeoffs of using gaze in more detail.

5.3 Generation Consistency

We analyze the generation performance of MemoLens using the llama-3.2-vision-instruct model. As discussed in §5.1.3, we use bert score to evaluate generation consistency. We inject the token merging layers into each layer in the vision encoder of the vision language model and prompt the model to generate captions. We then calculate and the BERT F1 score between the captions generated with no token merging and different token merging ratios. The results for two different models, the first based on LLaVa and the other based on LLama 3.2 architecture, are shown in Figure 9 (c). We see that the generated content is very similar to the original, scoring more than 88% even after 50% of the tokens are merged and more than 70% at a reduction ratio of 80% for both models.

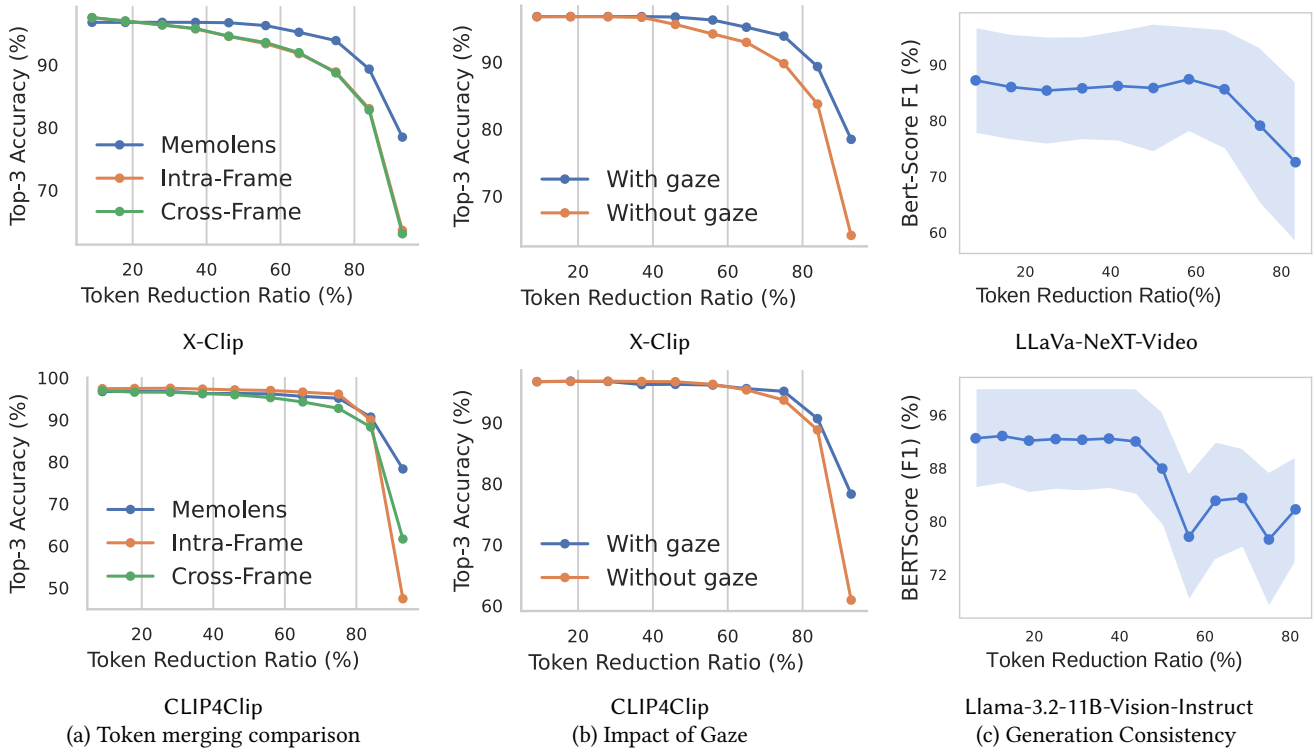
5.4 Search Efficiency

We analyze the scaling performance of MemoLens as we increase the number of samples in the data structure. For hierarchical search, we take the test set and organize it into trees of depth $L = 3$, where the top level (Level 2) corresponds to a script and the bottom level (Level 0) corresponds to the atomic activity. We randomly select n_2 nodes from Level 2, and for each node, we select n_1 child nodes corresponding to Level 1. Finally, for each of those Level 1 nodes, we select n_0 level 0 nodes. In total, we get n Level 0 tokens in a subset equal to $n_2 \times n_1 \times n_0$. We perform a hierarchical search on this tree structure following §3.3.1.

We plot the average time required for retrieval for 1000 queries as shown in Figure 10 in the smartphone. For the baseline, we can see that the time required for search increases with the total number of tokens. For the hierarchical search used in MemoLens, the number of computations is sub-linear on the total number of samples, making it much faster. For higher sample sizes, when the number of tokens increases significantly, the time for MemoLens will depend on n_2 instead of n. Since n_2 is $n_1 \times n_0$ times smaller than n, this is not significant. We chose a tree depth of $L = 3$ and a window size of $W = 8$ to balance speed with semantic preservation. In our evaluations, increasing depth to $L = 4$ causes over-compression of visual details, resulting in a 6% drop in top-3 retrieval accuracy. Thus, restricting the depth to $L = 3$ ensures that the abstract representations at the root level are sufficient to accurately route queries to their corresponding fine-grained details, while delivering the necessary speedup and limiting memory growth.

5.5 System Performance

To work on low-powered devices like smart glasses, MemoLens offloads the computation of the memory snippets to a personal device



(a) Token merging comparison (b) Impact of Gaze (c) Generation Consistency

Figure 9: Overall performance of MemoLens.

.

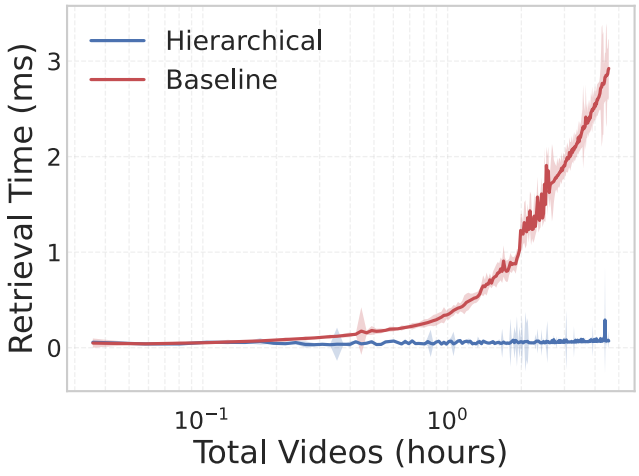
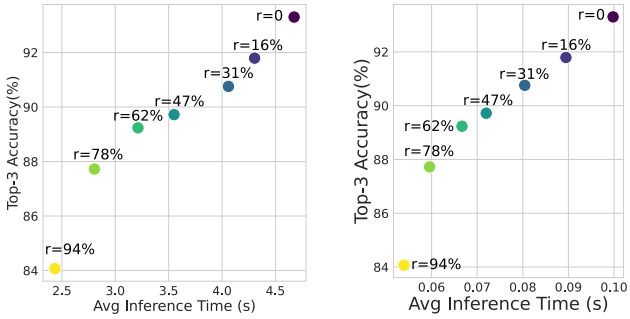


Figure 10: Retrieval latency of MemoLens.



(a) CPU (b) GPU

Figure 11: Performance of memory creation on cloud.

like a smartphone or external cloud. We consider two scenarios: 1) the user gives permission for their video data to be processed on the cloud, and 2) the user wants to keep their raw data in the device and only permits transfer of memory snippets to the cloud. In this case, the computation is done on a personal device like a smartphone. For simulating the cloud, we use a personal server with 4 A100 GPUs and an AMD EPYC 7513 32-Core Processor. We measure the average inference time for memory creation on both CPU and GPU.

For deploying on a smartphone CPU, we use a Galaxy S23 Ultra running Android v14. We export the model to ONNX format with opset version 20 to execute the model on the phone. Here, we also measure the inference time. Since portable devices will run on a battery, we also record the energy usage. The results of MemoLens in the two scenarios are discussed as follows:

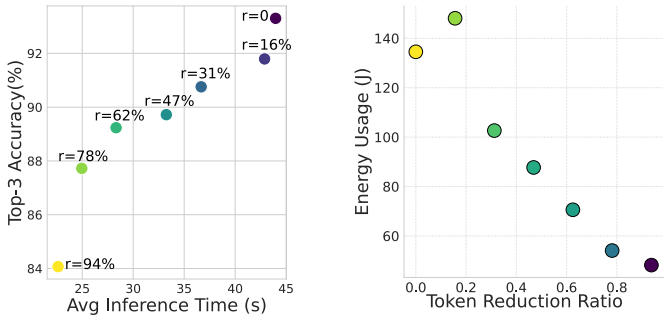


Figure 12: Performance of memory creation on mobile. The left figure shows the average inference time, and the right shows the energy usage.

Table 1: Peak memory usage in different platforms.

Token Reduction Ratio	Peak Memory Usage (MB)		
	Smartphone	CPU	GPU
0	4222	4789	6473
16%	4210	4803	6465
31%	4210	4774	6465
47%	4210	4772	6465
62%	4208	4770	6461
78%	4155	4767	6455
94%	**4108**	**4621**	**6443**

Performance in Cloud. Here, we consider the case when users opt to have their data processed on the cloud. We analyze the performance of MemoLens when processing and storing the compact memory in a server with and without a GPU. We measure the inference time for creating and storing one memory snippet, which corresponds to 8 frames or 8s of video. We run the system for 100 runs and take the average inference time in both CPU and GPU. We plot this against the top-3 retrieval accuracy. This is shown in Figure 11. From the figure, we see that the accuracy increases when we lower the token reduction ratio r. For desktop CPU, the system becomes real-time (i.e. inference time is lower than the duration of input) over a 31% token reduction ratio. The accuracy decreases with a higher token reduction ratio, while the inference time decreases. For GPU, the trends are similar; however, the system is much faster and operates in real-time in all cases. The results show that MemoLens can effectively speed up inference time and thereby server costs.

Performance in Smartphone. Now we consider the case when users want data to stay inside the device, and the computing is offloaded to a smartphone. We again analyze the performance of MemoLens when processing and storing the compact memory in a smartphone. We measure the inference time and energy usage for creating and storing one memory snippet, which corresponds to 8 frames or 8 seconds of video. We measure the inference time with the C++ chrono library and the energy usage via the Android Studio Profiler. We run the system for 10 runs with CLIP4Clip on smartphone CPU and take the average inference time in both. We plot this against the top-3 retrieval accuracy. This is shown in Figure 12. As shown, inference time decreases linearly from 43s to 21s when token reduction ratio increases from 16% to 94%. This pipeline does not need to keep up with wall-clock time strictly. It only needs to finish compressing the recording within a certain time budget. Inference can be done in batches across multiple cores in the background to decrease latency. We also plot the energy usage of MemoLens. The energy usage decreases with token reduction ratio. However, when token reduction ratio, $r = 0$, ie, no token merging is done, energy usage is 134J, which is lower than the energy usage of 148J at 16%. This increase is attributed to the computational overhead and memory movement costs of the token merging process itself, which outweigh the savings from reduced token count at low ratios. However, at higher reduction ratios, the reduced token count offsets this overhead, leading to significantly lower energy consumption. A reduced number of tokens offsets the higher energy usage at higher token reduction ratios, which is seen in the figure. As shown, MemoLens can execute much faster than the full model. Additionally, it consumes less energy at token reduction ratios above 16%. For a token reduction ratio of 62% and recording at 1fps, the battery would last approximately 7 hours. For optimal performance, the smartphone needs to optimize resource allocation between compute, energy and storage.

Memory Use in Different Platforms. We also measure the memory usage of MemoLens on different platforms. We show the peak memory use for server CPU, GPU, and smartphone for different token reduction ratios in Table 1. MemoLens requires slightly less memory in general compared to using full token sequence. The GPU requires more memory compared to other platforms, whereas the mobile requires the least. The lowest memory use is for 94% token reduction for all platforms, showing a 3.5% reduction for CPU, 2.7% for Mobile and 0.4% for GPU platforms.

Generation Latency on Cloud. To evaluate user experience, we analyze the latency of the generation with LLaVa-NeXT-Video hosted on the cloud server. We measure the Time to First Token (TTFT), which represents the duration between the cloud server receiving the retrieved context and generating the first token of the response. The distribution of TTFT is illustrated in Figure 13. The system demonstrates a median TTFT of 69ms. The latency distribution is tightly bounded, with the majority of requests served between 0.05s and 0.08s. The bimodal nature of the distribution (peaks around 55ms and 75ms), yet the latency remains well below standards of conversational UI (500 – 1000ms) with uplink time of 0.024s (considering 50% token merging ratio, top-3 memory snippets retrieved, and bandwidth of 500Mbps). This sub-second latency confirms that the cloud-based generation step does not introduce a perceptible bottleneck, maintaining real-time response for the end user.

Storage Use. We measure the storage requirements for the compact visual memory created by MemoLens. We show the storage requirements for saving two hours of video as memory snippets for different token reduction ratios in Table 2. The baseline is when there is no token reduction and hierarchical memory. The storage required is linearly dependent on the token reduction ratio.

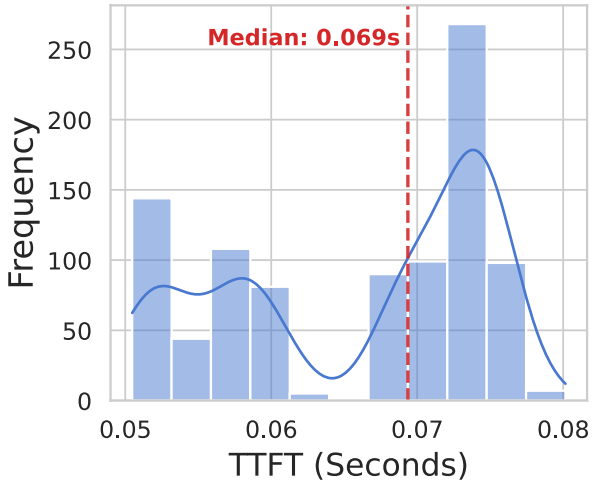


Figure 13: Text generation latency of MemoLens.

Table 2: Storage requirement for two hours of video in GB for different token reduction ratios.

	Baseline	MemoLens					
token reduction ratio	-	16%	31%	47%	62%	78%	94%
Storage (GB)	10.8	10.4	8.5	6.6	4.7	2.7	0.8

6 Related Work

Augmented Reality (AR) Systems. Our work is related to the research area of AR systems. Most of the existing work on AR systems [1, 8, 18, 21, 46, 49] focuses on optimizing system performance to reduce latency and power consumption. For example, NEAR [46] enables edge-based offloading of compute-intensive tasks such as object detection to reduce latency. Heimdall [49] manages multi-DNN GPU contention in mobile devices by decomposing neural network tasks, substantially improving frame rates and inference latency. FreeAR [1] introduces collaborative time slicing to distribute compute-intensive tasks across multiple devices, reducing power usage. Despite these advancements, these approaches share a common limitation: they optimize short-term computational efficiency but do not address the challenges of long-term visual memory retention or the effective management of continuous egocentric data. As a result, even when latency and inference performance improve, continuous recording generates large volumes of data that quickly overwhelm the limited storage and energy capacity of AR devices. In contrast, MemoLens focuses on the challenges of long-term visual memory retention and the storage bottlenecks inherent to continuous recording.

Eye Gaze. Eye Gaze is becoming more prominent in computer vision, offering insights about user focus and interactions. Studies have utilized gaze prediction to determine user attention areas within visual scenes [14, 29, 34, 35, 43] and analyze object interactions [27, 43, 45]. Recently, eye gaze has also been used to complement image and video modalities, in domains like action recognition [52], narration [9], and vision-language models [23]. Despite these advances, existing approaches primarily treat gaze as an auxiliary signal by using it to enhance recognition accuracy, rather than to fundamentally shape the representation of visual data. Current methods do not exploit gaze to reduce redundancy or guide efficient memory encoding, leaving the potential of gaze for long-term visual storage underexplored. In contrast, MemoLens introduces a novel use of gaze by directly integrating it into the tokenization process. This gaze-guided tokenization in MemoLens actively integrates gaze data into token merging, ensuring that stored representations are compact while preserving critical visual details.

Lifelogging and Memory Augmentation. Lifelogging refers to the continuous, digital recording of an individual's daily activities. Early work focused on wearable sensors to document daily life [15, 17, 19]. Recent AR-based systems like *NeverMind* [40] use spatial cues to facilitate memory encoding, but often focus on active learning rather than passive, continuous retrieval. More recent approaches, such as Byrne et al. [6], Le et al. [25], Shen et al. [41], encode visual experiences into language-based descriptions. However, translating rich egocentric video into text inevitably leads to the loss of fine-grained perceptual and contextual details. Furthermore, specialized systems like AIM [53] and TimeChat [38] optimize visual understanding in vision-language models (VLMs) via token merging, but primarily at the language decoder stage. Other recent works like Encode-Store-Retrieve (ESR) [41] and Memoro [54] utilize large language models to assist memory. While ESR's video-to-text conversion is computationally heavy and Memoro emphasizes auditory interfaces, MemoLens differentiates itself by storing visual data as compact, gaze-guided embeddings. By applying token merging at the initial visual encoding stage, MemoLens preserves richer information than language-only methods while maintaining the efficiency required for continuous edge-based lifelogging.

7 Conclusion

In this paper, we present the design, implementation, and evaluation of MemoLens, a spatial computing framework that empowers modern AR glasses with super memory capabilities. MemoLens addresses the key bottleneck of efficiently transforming and storing vast amounts of visual and gaze data into compact, retrievable memory representations suitable for AR devices. We implemented MemoLens and conducted a rich set of experiments using 100 hours of egocentric video data collected by real AR glasses. Our results show that MemoLens consistently outperforms the existing efficient video-language retrieval methods by achieving high accuracy and real-time retrieval even under substantial data compression. Therefore, we believe MemoLens represents a significant contribution to practical, always-on super memory capabilities in next-generation AR glasses. As our future work, we plan to extend MemoLens to incorporate additional sensor modalities like motion sensors and developing mechanisms to keep track of and delete outdated memory.

8 Acknowledgement

This study is supported in part by Meta Reality Lab and NSF Award NeTS-2312675.

References

[1] Kittipat Apicharttrisorn, Jiasi Chen, Vyas Sekar, Anthony Rowe, and Srikanth V Krishnamurthy. 2022. Breaking edge shackles: Infrastructure-free collaborative mobile augmented reality. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. Association for Computing Machinery, New York, NY, USA, 1–15.

[2] Robert Ariel and Alan D Castel. 2014. Eyes wide open: Enhanced pupil dilation when selectively studying important information. *Experimental Psychology: Learning, Memory, and Cognition* 40, 5 (2014), 1378–1391. Key finding: Pupil dilation and gaze duration increase for items people are incentivized to remember..

[3] Ajay Sudhir Bale, Naveen Ghorpade, Muhammed Furqaan Hashim, Jatin Vaishnav, and Zahra Almaspoor. 2022. A comprehensive study on metaverse and its impacts on humans. *Advances in Human-Computer Interaction* 2022, 1 (2022), 3247060.

[4] Michael Barz, Sebastian Kapp, Jochen Kuhn, and Daniel Sonntag. 2021. Automatic recognition and augmentation of attended objects in real-time using eye tracking and a head-mounted display. In *ACM Symposium on Eye Tracking Research and Applications*. Association for Computing Machinery, New York, NY, USA, 1–4.

[5] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token Merging: Your ViT But Faster. In *ICLR*. OpenReview.net, Amherst, MA, USA, 15 pages.

[6] Daragh Byrne, Aiden R Doherty, Cees GM Snoek, Gareth JF Jones, and Alan F Smeaton. 2010. Everyday concept detection in visual lifelogs: validation, relationships and trends. *Multimedia Tools and Applications* 49, 1 (2010), 119–144.

[7] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. Extracting Training Data from Diffusion Models. In *Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, USA, 5233–5250.

[8] Kaifei Chen, Tong Li, Hyung-Sin Kim, David E Culler, and Randy H Katz. 2018. Marvel: Enabling mobile augmented reality with low energy and low latency. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. Association for Computing Machinery, New York, NY, USA, 292–304.

[9] Xianyu Chen, Ming Jiang, and Qi Zhao. 2024. Gazexplain: Learning to predict natural language explanations of visual scanpaths. In *European Conference on Computer Vision*. Springer, Springer, Cham, Switzerland, 314–333.

[10] Dima Damen et al. 2018. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In *European Conference on Computer Vision (ECCV)*. Springer, Cham, Switzerland, 720–736.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. OpenReview.net, Amherst, MA, USA, 22 pages. https://openreview.net/forum?id=YicbFdNTTy

[12] Alexey Dosovitskiy and Thomas Brox. 2016. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, Los Alamitos, CA, USA, 4829–4837.

[13] David Dzsotjan, Kim Ludwig-Petsch, Sergey Mukhametov, Shoya Ishimaru, Stefan Küchemann, and Jochen Kuhn. 2021. The predictive power of eye-tracking data in an interactive AR learning environment. In *Adjunct proceedings of the 2021 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2021 ACM international symposium on wearable computers*. Association for Computing Machinery, New York, NY, USA, 467–471.

[14] Yini Fang, Jingling Yu, Haozheng Zhang, Ralf van der Lans, and Bertram Shi. 2024. OAT: Object-level attention transformer for gaze scanpath prediction. In *European Conference on Computer Vision*. Springer, Springer, Cham, Switzerland, 366–382.

[15] Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Zhu. 2002. MyLifeBits: Fulfilling the Memex Vision. In *Proceedings of the 10th ACM International Conference on Multimedia*. Association for Computing Machinery, New York, NY, USA, 235–238.

[16] Kristen Grauman, Andrew Westbury, and et al. 2022. Ego4D: Around the World in 3,000 Hours of Egocentric Video. In *CVPR*. IEEE, Los Alamitos, CA, USA, 16142–16153.

[17] Cathal Gurrin, Alan F Smeaton, and Aiden R Doherty. 2014. Lifelogging: Personal Big Data. *Foundations and Trends in Information Retrieval* 8, 1 (2014), 1–125.

[18] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2014. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. Association for Computing Machinery, New York, NY, USA, 68–81.

[19] Morgan Harvey, Marc Langheinrich, and Geoff Ward. 2016. Remembering through lifelogging: A survey of human memory augmentation. *Pervasive and Mobile Computing* 27, 1 (2016), 14–26.

[20] Mary Hayhoe and Dana Ballard. 2005. Eye movements in natural behavior. *Trends in Cognitive Sciences* 9, 4 (2005), 188–194. Key finding: Gaze is proactive and goal-driven, focusing on information required for current or future tasks..

[21] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2015. Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. Association for Computing Machinery, New York, NY, USA, 331–344.

[22] Sebastian Kapp, Michael Barz, Sergey Mukhametov, Daniel Sonntag, and Jochen Kuhn. 2021. ARETT: Augmented reality eye tracking toolkit for head mounted displays. *Sensors* 21, 6 (2021), 2234.

[23] Robert Konrad, Nitish Padmanaban, J Gabriel Buckmaster, Kevin C Boyle, and Gordon Wetzstein. 2024. Gazegpt: Augmenting human capabilities using gaze-contingent contextual ai for smart eyewear. arXiv preprint arXiv:2401.17217.

[24] Amel Ksibi, Ala Saleh D Alluhaidan, Amina Salhi, and Sahar A El-Rahman. 2021. Overview of lifelogging: current challenges and advances. *IEEE Access* 9, 1 (2021), 62630–62641.

[25] Huy Viet Le, Sarah Clinch, Corina Sas, Tilman Dingler, Niels Henze, and Nigel Davies. 2016. Impact of video summary viewing on episodic memory recall: Design guidelines for video summarizations. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. Association for Computing Machinery, New York, NY, USA, 4793–4805.

[26] Yin Li, Mengmeng Liu, and James M. Rehg. 2018. In the Eye of Beholder: Joint Learning of Gaze and Actions in First Person Video. In *ECCV*. Springer, Cham, Switzerland, 326–341.

[27] Zhi-Yi Lin, Jouh Yeong Chew, Jan van Gemert, and Xucong Zhang. 2024. Gaze-HTA: End-to-end Gaze Target Detection with Head-Target Association. arXiv preprint arXiv:2404.10718.

[28] Huaishao Luo, Lei Ji, Botian Shi, and et al. 2022. CLIP4Clip: An Empirical Study of CLIP for End-to-End Video Clip Retrieval. *Neurocomputing* 508, 1 (2022), 263–273.

[29] Mathias N Lystbæk, Ken Pfeuffer, Tobias Langlotz, Jens Emil Sloth Grønbæk, and Hans Gellersen. 2024. Spatial Gaze Markers: Supporting Effective Task Switching in Augmented Reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–11.

[30] Lingni Ma, Yuting Ye, Fangzhou Hong, Vladimir Guzov, Yifeng Jiang, Rowan Postyeni, Luis Pesqueira, Alexander Gamino, Vijay Baiyya, Hyo Jin Kim, Kevin Bailey, David Soriano Fosas, C. Karen Liu, Ziwei Liu, Jakob Engel, Renzo De Nardi, and Richard Newcombe. 2024. Nymeria: A Massive Collection of Multimodal Egocentric Daily Motion in the Wild. arXiv:2406.09905 [cs.CV] https://arxiv.org/abs/2406.09905

[31] Yiwei Ma, Guohai Xu, Xiaoshuai Sun, Ming Yan, Ji Zhang, and Rongrong Ji. 2022. X-clip: End-to-end multi-grained contrastive learning for video-text retrieval. In *Proceedings of the 30th ACM international conference on multimedia*. Association for Computing Machinery, New York, NY, USA, 638–647.

[32] Paul Milgram and Fumio Kishino. 1994. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems* 77, 12 (1994), 1321–1329.

[33] Bipul Mohanto, A. B. M. Tariqul Islam, Enrico Gobbetti, and Oliver Staadt. 2022. An integrative view of foveated rendering. *Computers & Graphics* 102, 1 (2022), 474–501. https://doi.org/10.1016/j.cag.2021.10.010

[34] Sounak Mondal, Seoyoung Ahn, Zhibo Yang, Niranjan Balasubramanian, Dimitris Samaras, Gregory Zelinsky, and Minh Hoai. 2024. Look Hear: Gaze Prediction for Speech-directed Human Attention. In *European Conference on Computer Vision*. Springer, Springer, Cham, Switzerland, 236–255.

[35] Süleyman Özdel, Yao Rong, Berat Mert Albaba, Yen-Ling Kuo, Xi Wang, and Enkelejda Kasneci. 2024. A Transformer-Based Model for the Prediction of Human Gaze Behavior on Videos. In *Proceedings of the 2024 Symposium on Eye Tracking Research and Applications*. Association for Computing Machinery, New York, NY, USA, 1–6.

[36] Toby Perrett, Ahmad Darkhalil, Saptarshi Sinha, Omar Emara, Sam Pollard, Kranti Kumar Parida, Kaiting Liu, Prajwal Gatti, Siddhant Bansal, Kevin Flanagan, et al. 2025. Hd-epic: A highly-detailed egocentric video dataset. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE, Los Alamitos, CA, USA, 23901–23913.

[37] Chiara Plizzari, Gabriele Goletto, Antonino Furnari, Siddhant Bansal, Francesco Ragusa, Giovanni Maria Farinella, Dima Damen, and Tatiana Tommasi. 2024. An outlook into the future of egocentric vision. *International Journal of Computer Vision* 132, 11 (2024), 4880–4936.

[38] Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. 2024. Timechat: A time-sensitive multimodal large language model for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Los Alamitos, CA, USA, 14313–14323.

[39] Ricardo Ribeiro, Alina Trifan, and António JR Neves. 2022. Lifelog retrieval from daily digital data: narrative review. *JMIR mHealth and uHealth* 10, 5 (2022), e30517.

[40] Oscar Rosello Gil Rosello. 2017. *NeverMind: an interface for human Memory augmentation*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[41] Junxiao Shen, John J. Dudley, and Per Ola Kristensson. 2024. Encode-Store-Retrieve: Augmenting Human Memory through Language-Encoded Egocentric Perception . In *2024 IEEE International Symposium on Mixed and Augmented*

Reality (ISMAR). IEEE Computer Society, Los Alamitos, CA, USA, 923–931. https://doi.org/10.1109/ISMAR62088.2024.00108

[42] Congzheng Song and Ananth Raghunathan. 2020. Information Leakage in Embedding Models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 347–361. https://doi.org/10.1145/3372297.3417270

[43] Florian Strohm, Mihai Bâce, and Andreas Bulling. 2024. Learning user embeddings from human gaze for personalised saliency prediction. *Proceedings of the ACM on Human-Computer Interaction* 8, ETRA (2024), 1–16.

[44] Benjamin W Tatler, Mary M Hayhoe, Michael F Land, and Dana H Ballard. 2011. Eye guidance in natural vision: Reinterpreting salience. *Journal of vision* 11, 5 (2011), 5–5.

[45] Jie Tian, Ran Ji, Lingxiao Yang, Suting Ni, Yuexin Ma, Lan Xu, Jingyi Yu, Ye Shi, and Jingya Wang. 2024. Gaze-guided Hand-Object Interaction Synthesis: Dataset and Method. arXiv preprint arXiv:2403.16169.

[46] Marco Trinelli, Massimo Gallo, Myriana Rifai, and Fabio Pianese. 2019. Transparent AR processing acceleration at the edge. In *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*. Association for Computing Machinery, New York, NY, USA, 30–35.

[47] Jianghao Xiong, En-Lin Hsiang, Ziqian He, Tao Zhan, and Shin-Tson Wu. 2021. Augmented reality and virtual reality displays: emerging technologies and future perspectives. *Light: Science & Applications* 10, 1 (2021), 216.

[48] Jingkang Yang, Shuai Liu, Hongming Guo, Yuhao Dong, Xiamengwei Zhang, Sicheng Zhang, Pengyun Wang, Zitang Zhou, Binzhu Xie, Ziyue Wang, et al. 2025. Egolife: Towards egocentric life assistant. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE, Los Alamitos, CA, USA, 28885–28900.

[49] Juheon Yi and Youngki Lee. 2020. Heimdall: mobile GPU coordination platform for augmented reality applications. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. Association for Computing Machinery, New York, NY, USA, 1–14.

[50] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*. OpenReview.net, Amherst, MA, USA, 11 pages. https://openreview.net/forum?id=SkeHuCVFDr

[51] Yuanhan Zhang, Bo Li, haotian Liu, Yong jae Lee, Liangke Gui, Di Fu, Jiashi Feng, Ziwei Liu, and Chunyuan Li. 2024. LLaVA-NeXT: A Strong Zero-shot Video Understanding Model. https://llava-vl.github.io/blog/2024-04-30-llava-next-video/

[52] Zehua Zhang, David Crandall, Michael Proulx, Sachin Talathi, and Abhishek Sharma. 2022. Can gaze inform egocentric action recognition?. In *2022 Symposium on Eye Tracking Research and Applications*. Association for Computing Machinery, New York, NY, USA, 1–7.

[53] Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. 2025. Aim: Adaptive inference of multi-modal llms via token merging and pruning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. IEEE, Los Alamitos, CA, USA, 20180–20192.

[54] Wazeer Deen Zulfikar, Samantha Chan, and Pattie Maes. 2024. Memoro: Using large language models to realize a concise interface for real-time memory augmentation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–18.