ChatGPT

Design of a Couples Therapy Chat System with DeepSeek-OCR Optical Memory

System Architecture Overview

The proposed system is a **multimodal LLM-based chat agent** augmented with an optical long-term memory mechanism. It integrates *DeepSeek-OCR* for compressing and storing long dialog histories as images (optical snapshots) and an LLM for real-time conversation generation. The high-level architecture consists of several components:

- **User Interface & Input Handler:** Accepts **turn-based inputs** (text or images) from each participant. This layer formats user messages and images into a standardized internal request.
- **Multimodal LLM (Therapy Agent):** The core **conversation model** (e.g. a fine-tuned large language model) capable of processing both text tokens and image-based tokens. It incorporates a vision encoder (DeepSeek's *DeepEncoder* or similar) for image inputs and a language decoder for text generation [1] [2]. The LLM is fine-tuned for therapeutic dialog, ensuring empathetic and coherent responses.
- **DeepSeek-OCR Compression Pipeline:** A specialized subsystem for **optical memory compression**. It renders textual transcripts into image form and uses DeepSeek's **DeepEncoder** to convert these images into compact **vision tokens** [3] [4]. This pipeline is responsible for compressing long text histories into images efficiently.
- **Short-Term Memory Buffer:** A fast access buffer storing the **recent dialogue turns** in plain text. This covers the most recent context (e.g. the last few exchanges) that remains uncompressed for full fidelity.
- **Optical Long-Term Memory Store:** A database of **compressed conversation history** in image form (with associated metadata). Older turns are stored as rendered images (optical snapshots) rather than raw text, dramatically reducing token count for storage [5] [6]. Each stored image may represent a segment of the dialogue (e.g. a few thousand tokens of text compressed into one image).
- **Memory Manager & Forgetting Module:** Orchestrates what content stays in short-term memory vs. what gets compressed into long-term **optical memory**. It applies a *memory decay policy* (inspired by the *forgetting curve*) that progressively compresses or summarizes older context over time [6] [7]. This component triggers the DeepSeek-OCR pipeline for compression and optionally generates text summaries for very old content.
- **Retrieval & Reactivation Module:** Handles **retrieving past dialog** from the optical memory store when needed. It can search metadata or embeddings of past turns to identify relevant segments, then **reactivate** those by feeding their stored images back through the vision encoder into the LLM's context. This allows the model to recall specific past details on-demand from the compressed memory [8] [9].
- **vLLM Inference Engine:** The serving infrastructure (built on *vLLM*) that **hosts the LLM model** and manages efficient execution. vLLM's optimized scheduler and *PagedAttention* memory management enable the system to handle **very long contexts** and multiple conversations efficiently [10] [11]. It

ensures real-time responses by batching requests and only allocating GPU memory to active context pages, which is crucial for long dialogues with many memory tokens.

These components work together to maintain an **ultra-long conversational context** in a resource-efficient way. The design cleanly separates the conversational logic (handled by the LLM) from the memory extension (handled by the DeepSeek-OCR pipeline and memory manager).

Message Processing Workflow

To illustrate how a turn is processed and how the memory system works, consider a single turn in the therapy dialog:

1. **User Input Reception:** A user (partner A or B) sends a message. This could be a text message or an image (e.g. a photo or a diagram they share). The input handler tags it with a turn ID and timestamp.
2. **Input Preprocessing:**
3. If the input is **text**, it's passed through normally (with minimal preprocessing like truncating extremely long single messages, if needed).
4. If the input is an **image**, the system invokes the DeepSeek *vision encoder* (or a generic vision module if it's not a text image) to produce an embedding or description. For example, if a user shares a photo, the system can run an OCR or image captioning as appropriate. (For text-heavy images, DeepSeek's OCR capability is used to convert it into text or tokens for the LLM.)
5. **Context Assembly:** The **prompt context** for the LLM is assembled. This includes:
6. The latest few exchanges from the **Short-Term Memory Buffer** in plaintext (to provide immediate context in full detail).
7. **Relevant long-term memory** images, if any: The retrieval module checks the new user query for references or topics that might relate to older discussion. It searches the metadata/embeddings of stored memory chunks. If a relevant past segment is found (e.g. a past session discussing a similar argument), its corresponding image is fetched from the Optical Memory Store. That image is then fed through the DeepEncoder to produce compressed vision tokens representing that past conversation, which are included in the LLM's context window [8] [9]. (In a unified multimodal model, the image itself can be provided to the model as an input alongside text.)
8. A **system prompt or role prompt** (if any) that defines the therapist agent's persona and guidelines (this is static and reused every turn).
9. **Offline knowledge or tools (if needed):** In an offline analysis mode, the system could also include summaries or analytical notes from previous sessions, but in real-time mode this is usually omitted to save context space.
10. **LLM Response Generation:** The assembled context (mix of current text, memory images as vision tokens, etc.) is fed into the LLM via the vLLM engine. The LLM then generates a response (the therapist's reply) conditioned on both the textual and optical context. Because the model can "see" the images, it effectively has access to far more dialogue history than a text-only context would allow. For example, it can *read* an image containing a summary of a session from last month if that was retrieved, and incorporate that knowledge when forming its answer.
11. **Output Post-processing:** The LLM's answer (text) is returned. The system may perform mild post-processing (e.g. ensuring it's within length limits or doesn't violate any safety rules), then delivers the response to the users.
12. **Memory Update:** After the turn, the new user message and the assistant's reply are added to the **Short-Term Memory Buffer** (as plain text). The memory manager then evaluates the buffer length:

13. If the recent conversation exceeds a certain length threshold (e.g. beyond the model's high-precision window or a configured token limit), the oldest turns in the short-term buffer are **removed and compressed**. The removed transcript segment is rendered into an image (see next section for details) and stored in the **Optical Memory Store**, indexed by turn range and timestamps. A reference to this new "memory snapshot" is kept (with embeddings for semantic search).
14. Optionally, an **abstractive summary** of that segment is generated (either by the LLM or a smaller model) and stored alongside the image as metadata. This helps in quickly searching memory by topic without decoding every image each time.
15. **Offline Processing (if applicable):** In off-peak times or after a session ends, the system can perform batch operations: e.g. re-compress multiple image memories further (combine several images into a single more compressed image), run periodic summarization on long-term logs, or train/update the retrieval index. These offline jobs optimize the memory store without impacting live chat latency.

This workflow ensures that at each turn the LLM has the **most pertinent context** (recent turns in detail and older relevant turns in compressed form) to generate a coherent and contextually informed response. The division between real-time updates and offline maintenance keeps the conversation flowing smoothly while slowly **aging out** old details into efficient storage.

Optical Memory Compression Pipeline

A key innovation is the **DeepSeek-OCR-based compression** of the conversation history into images, effectively creating an "optical memory." The pipeline for rendering chat history into DeepSeek-compatible vision inputs works as follows:

- **Chunking:** The text transcript is divided into chunks (e.g. by conversation episodes or by token count). For instance, the system might bundle ~1000 words of dialogue (which could be several thousand text tokens) into one image chunk [12] [13]. Chunking can be done at conversation breaks (e.g. end of a session or topic) or simply by size to ensure images are a manageable resolution.
- **Rendering to Image:** Each text chunk is rendered as an image (like a screenshot of a chat log). The rendering follows **format guidelines** optimized for OCR:
- Use a **clear, high-contrast font** (e.g. black sans-serif text on white background) to maximize OCR accuracy.
- Arrange text in a **single column with appropriate wrapping** so that lines are not too long. Multi-column or very dense text can reduce accuracy, so a consistent readable layout is chosen.
- Choose an **image resolution** mode based on the importance of the content. DeepSeek-OCR supports multiple resolution "modes" (e.g. 512×512, 640×640, 1024×1024, 1280×1280) with corresponding numbers of vision tokens [14] [15]. For *high-fidelity chunks* (important recent history), the system uses a higher resolution (larger image, meaning more pixels and thus more vision tokens) to capture details. For less critical or very old chunks, a smaller image (lower resolution) may be used to save space. For example, a recent conversation chunk might be rendered at 1024×1024, whereas an older one might be rendered at 640×640 or with aggressive downsampling.
- **Compression vs. readability trade-off:** The font size and spacing can be adjusted to fit more text into the image (increasing compression ratio), but there's a limit – if text is too small or cramped, the OCR decoder might start to miss words. In practice, DeepSeek-OCR achieves ~97% text recovery accuracy when text tokens are compressed by <10× relative to vision tokens [16]. So the system targets compression ratios in that range for important content. It may allow higher compression (15–20×) for very old content, accepting some loss of fidelity for the sake of brevity [17].

- **DeepEncoder Processing:** The rendered image is then passed through DeepSeek's **DeepEncoder**. This vision encoder (≈380M parameters) will split the image into patches and produce a set of **latent vision tokens** capturing the text content [3] [18]. Thanks to the specialized architecture (using local window attention + token downsampling), the encoder yields a highly compressed token sequence (e.g. 256 vision tokens for a 1024×1024 image, after a 16× token compression) [19]. These vision tokens act as the *compressed representation* of the entire chunk of dialogue.
- **Storage in Memory Store:** The system stores the **image file** (for potential future use by other tools or verification) along with the encoded form:
- If the LLM and system support storing **precomputed vision tokens**, it could store the DeepEncoder's output (the 256 tokens) directly in the memory database to avoid re-encoding the image each time. These could be saved as numeric arrays.
- Alternatively, if storing raw tokens is impractical, the image path is stored, and on retrieval the image will be re-encoded on the fly. (Caching strategies can be used to avoid repeating the encoding too often for frequently accessed memories.)
- **Token Count Advantage:** This optical compression provides a **dramatic reduction in context length**. A chunk that might have been thousands of text tokens is now on the order of a few hundred vision tokens [5]. The LLM can thus attend to very long histories by trading off some textual fidelity for far fewer tokens. Experiments from DeepSeek-OCR show that even a full page of dense text can be encoded with ~100 vision tokens at 10× compression with minimal loss [12] [13]. This enables the conversation to span *hundreds of pages* of dialogue if needed, far beyond standard context limits [20].
- **Handling Images in Dialogue:** If the user's turn itself is an **image (non-textual)**, the system can also leverage the vision encoder. For example, if a user shows a handwritten letter or an infographic, the DeepEncoder can process it similarly to extract text. Non-text content (like a photo of the couple) might require a different vision model or the same encoder if it was trained on general images as well [21] [22]. The architecture can accommodate this by either using DeepSeek-OCR's generalized capabilities (it was trained with some general vision data) or falling back to another vision model for purely visual understanding. In either case, image inputs are converted to a token representation that the language model can incorporate.

By following these steps, the system **compresses the dialogue history into a compact visual form**. The format and resolution for each snapshot can be tuned dynamically: e.g., use higher DPI or larger images for key moments (to ensure near-lossless recall) and smaller, more compressed images for peripheral or repetitive dialogue. The result is an efficient memory of the conversation that the LLM can decode later when needed, without having to carry the full text of the history at all times.

Memory Decay and the Forgetting Curve Mechanism

To manage ultra-long conversations, the system implements **memory decay** – gradually reducing the fidelity of stored context – inspired by DeepSeek-OCR's analogy to the human *forgetting curve*. Recent dialogue is preserved with high detail, whereas older dialogue is retained only in summary or low-detail form, emulating how human memory fades over time [6] [7]. This not only saves resources but also prioritizes important information.

Memory decay mechanism in the optical memory system: recent turns are stored as clear, high-resolution images (high fidelity), while older turns "fade" into blurrier, low-resolution snapshots (lower information density) [23]. This mimics the human forgetting curve, preserving essential context while freeing up memory.

Concretely, the **Memory Manager** applies policies such as:

- **Full Detail for Recent Turns:** The latest dialogue (e.g. the current session's conversation or last N turns) remains in text form or high-resolution image form. The LLM sees these with full clarity. This is akin to short-term memory where details are sharp.
- **Partial Compression for Intermediate Memory:** As the conversation grows, intermediate-age content (e.g. earlier in the day or previous session) is compressed to images but with **moderate resolution**. The text isn't in the immediate context anymore, but when needed the model can decode these images to recall specifics. The compression might be ~5–10× for these, maintaining ~90–97% of details [17]. In effect, the system "forgets" exact wording but retains semantic content and who said what.
- **High Compression for Old Memory:** Very old or less important context is stored in highly compressed form – **small or blurred images** that capture only the gist. This could be 15–20× compressed, where only key words or layout are recoverable reliably. The idea is that the model retains an *impression* of these events, not exact dialogue [24] [6]. For example, a fight the couple had a year ago might be referenced only abstractly unless specifically needed.
- **Summarization:** In addition to images, the system can store a short **textual summary** of old conversations (a few sentences) in the long-term store. This acts like an "index" or semantic memory of what happened, without details. The LLM can always access this summary as text (low cost) and only delve into the image if finer detail is required.
- **Memory Refresh of Important Details:** If a certain past topic becomes relevant again (e.g. one partner keeps bringing up *"the vacation argument from last June"*), the system can **reactivate or re-encode that memory at higher fidelity**. Inspired by human recall (where recalling a memory can reinforce it), the memory manager might take an old blurry snapshot and re-render it in high resolution or even restore it to text form for the current context. In practice, this could mean re-processing the original text log (if available) or upsampling the stored image and running it through OCR to get the text back, then giving that text to the LLM. This *refresh* mechanism ensures that important long-term facts aren't lost to compression – they can be **brought back into focus on demand** [25] [26].
- **Forgetting Irrelevant Data:** If certain details never get used or referenced, they remain in compressed form and eventually could be purged or archived offline. This keeps the working memory lean. For example, small talk or resolved tangents from many sessions ago might never be needed again and could be discarded after being summarized.

This controlled forgetting not only saves computation but also aligns with cognitive principles. By not remembering every word indefinitely, the agent avoids drowning in irrelevant details, focusing instead on *meaningful patterns and emotional themes* from the couple's history. The DeepSeek-OCR approach provides the tooling to implement this: it quantifies how much detail is preserved at each compression level (e.g. 97% at 10×, 60% at 20×) [17], so the system can dial memory fidelity up or down as needed. The figure above illustrates this concept of decaying memory: fresh memories are crystal-clear, and older ones gradually blur out – yet **critically, they are never entirely lost**, only minimized until needed [27] [7].

Context Retrieval and Reactivation of Past Dialog

Storing conversation history as images would be futile without a way to **retrieve and use** those memories. The system therefore includes a retrieval mechanism to decide *when and which* compressed memories to bring back into the LLM's context.

Memory Indexing: Every time a chunk of conversation is compressed into an image, the system records metadata: - Timestamps (e.g. this image covers chat from Jan 5, 2025, 10:00–10:30). - Participants and topics (e.g. tags like *"discussed trust issues", "mentioned Alice's mother"* if such keywords appear, which could be extracted via a quick NLP pass before compression). - An **embedding** of the chunk's text (if the text is available pre-compression, the system uses a sentence transformer or the LLM itself to get a vector embedding representing the content's semantics).

These metadata are stored in a vector database or index. This allows semantic **search** over the long-term memory: when a new user query arrives, the retrieval module can find which past segments are most likely relevant. For example, if the user says, *"I feel we're repeating the conversation we had about finances last month,"* the system can search the index for "finances" and find that chunk from last month.

Retrieval Process: At each user turn (or when the assistant is about to respond), the system can do the following: - Use the latest user utterance (and possibly a few recent turns for context) as a **query** into the memory index. This could be keyword-based or embedding-based. The result is a ranked list of candidate memory chunks that might inform the current turn. - Apply rules to filter the candidates: e.g., limit to the top 1 or 2 most relevant memory images to avoid overloading context. Also, ensure we don't retrieve something the user explicitly asked the agent to forget or that might be sensitive to resurface inappropriately (therapeutic context might require some discretion). - For each selected memory chunk, fetch the corresponding **image** from the store. If the system cached the DeepEncoder output tokens for it, those can be fetched directly instead (saving having to load and re-encode the image). - Insert these memory tokens (or images) into the LLM's context, along with a brief marker. For instance, in the prompt you might have: "*(Recalled memory from <date>:)*" followed by either the image (if the model can handle it directly) or by a special token sequence that the model knows to interpret via its vision encoder. In an implementation, one might actually feed the image as an input to the multimodal model, or if using a pipeline, run the image through DeepSeek's decoder to get text and prepend it as a quoted transcript in the prompt (though the latter sacrifices some token advantage).

Integration of Textual + Optical Context: The model now receives a mix of *explicit text* (recent conversation, system instructions, etc.) and *vision tokens* representing older dialog. During generation, the LLM attends over all these. Ideally, the **language model was trained or fine-tuned to handle this multimodal context**, so it can seamlessly use the decoded content of the images. (DeepSeek-OCR's decoder was a 3B MoE specialized for OCR [28], but for a larger therapy model, one might integrate the vision encoder and train the combined system to treat image tokens as just another part of context.)

At generation time, the LLM can thus *"remember"* details from far back by effectively reading those images. For example, if the user asks *"Why do you keep saying that?"* the assistant can look at a recalled image of last session's transcript to see what it had said before that might be similar. The LLM's answer will incorporate that memory (e.g. *"Last session, I observed that you mentioned a similar feeling: 'I always say X when Y happens'…").*

Reactivation and Re-compression: After the turn, a post-processing can occur: - If a memory image was used and found helpful, the system can mark it as *"recently accessed"*. This could prompt the memory manager to keep it at higher fidelity (perhaps don't further downsample it in the near future, or even regenerate a fresh image with larger resolution if it's likely to be used again). - If new topics emerged that will likely be referenced, the system might proactively retrieve related memories in the next turn or keep them ready in a cache.

This retrieval mechanism ensures **past context is not lost** in the optical memory. Even though it's not in the working memory by default, it can be *selectively restored*. By combining textual short-term memory and visual long-term memory, the agent achieves a form of **context reconstruction**: when needed, the original dialogue can be reconstructed (in pieces) from the compressed form and provided to the model. In effect, the architecture works like an externalized memory for the LLM, akin to how a human might check an old diary or notes to recall what was said – here the LLM "reads" the archived dialog via OCR.

LLM Integration and vLLM Deployment

Model Architecture: The chat system uses a **multimodal LLM** that is extended to accept image inputs alongside text. One approach is to adopt the DeepSeek-OCR architecture itself as the backbone: it has a DeepEncoder + MoE decoder that already is designed to output text from images [2] [28]. However, the decoder (3B MoE) used in DeepSeek-OCR might be limited in conversational ability. In practice, we can merge this vision encoder with a larger pre-trained LLM (like LLaMA-2 or GPT-4 architecture) that has been fine-tuned on counseling dialogue. This could be done by attaching the DeepEncoder to the transformer backbone of a conversational model, allowing the model's self-attention to attend over both text tokens and the vision tokens from images. During fine-tuning, the model learns to interpret those vision tokens as just compressed text. (For example, the model could be trained on synthetic conversations where part of the context was provided as an image and it had to answer questions about it, thereby learning to decode the optical tokens.)

If integrating at architecture-level is too complex, a simpler pipeline could be: - Use DeepSeek-OCR as a **preprocessor**: whenever an image memory needs to be read, run the DeepSeek decoder model to get the text content, then feed that text into the main LLM. This would allow using any existing LLM as is. The downside is losing some token efficiency (since you convert back to text tokens), but it guarantees the main model understands the content. Depending on context length, this might still be viable for moderate lengths and simplifies integration. - In our design, however, we assume a more *seamless integration* where the main model itself can handle images (since models like GPT-4 have demonstrated this ability). This way, memory images can be fed directly without an intermediate text expansion, preserving the token savings.

vLLM Optimization: The system is deployed on top of the **vLLM inference engine** for performance. vLLM offers several advantages for this architecture [29] [10]:

- *PagedAttention for Long Contexts:* vLLM's memory management allocates KV-cache pages dynamically and only for active tokens [30] [10]. This means even if the conversation history grows extremely large (many thousands of tokens, including vision tokens), the GPU memory isn't overwhelmed by one huge contiguous context. Inactive parts of the context (e.g. tokens from much earlier in the prompt that are not currently needed in attention) can be offloaded. This allows **practical handling of long dialogs** without running out of memory or sacrificing speed.
- *Continuous Batching:* In a live therapy chat, response latency is important. vLLM's continuous batching ensures incoming messages are processed with minimal waiting, improving real-time responsiveness [31] [32]. Even if multiple therapy sessions run on one server, vLLM intermixes their token generation to fully utilize the GPU, achieving higher throughput and lower latency.
- *Memory Reuse:* Many portions of the prompt remain constant or get repeated (for instance, the system persona, or if the user repeats a phrase from memory that's provided via an image every time it's recalled). vLLM can reuse identical prefix KV caches across requests. Thus, if our agent

always includes certain memory images or the same system prompt, those don't need to be recomputed each turn – improving efficiency.
- *Scalability:* The design can support **multiple concurrent conversations** (multiple couples) by leveraging vLLM's multi-tenant support [33]. The optical memory approach actually aids scalability too: since most of the old context is compressed, each session uses fewer tokens on average, allowing more sessions to fit on one GPU server. vLLM further ensures that GPU memory is only occupied by active tokens of active sessions [10], so a paused or slow conversation's long-term memory isn't tying up resources.

In summary, vLLM's infrastructure complements the architecture by making **long-context handling feasible in production**. The LLM itself can be larger (billions of parameters) but thanks to quantization and efficient scheduling (both supported by vLLM), we can deploy the system cost-effectively [34] [35].

Latency Considerations: Incorporating vision encoding/decoding has some overhead. DeepSeek's encoder is fairly light (~380M, a fraction of the decoder size) and can run quickly on a GPU [36]. Still, encoding a 1024×1024 image into tokens is an extra step that text-only models don't do. To mitigate this: - The system can pre-encode images asynchronously (e.g. right when a memory chunk is created, store its tokens). - During retrieval, fetching and using cached vision tokens avoids rerunning the encoder. - If using the integrated model, the image encoding is part of the model's forward pass and is optimized just like text tokens – given the 10× reduction in token count, it's usually worth the small cost of the vision encoder to avoid the cost of many more text tokens [5] [37]. - vLLM's batching could even batch multiple image encodings together if using separate calls, to further accelerate throughput.

Practical Implementation Notes and Trade-offs

Designing and deploying this system entails some important **trade-offs and considerations**:

- **Accuracy vs Compression:** There is a balance between how much we compress the history and how faithfully it can be reconstructed. At ~10× compression, DeepSeek-OCR is nearly lossless for text [17], but at 20× compression, a significant amount (up to 40%) might be lost or misread [24]. In a therapy setting, losing details could be problematic (misremembering what someone said could break trust). Thus, the system should compress conservatively for critical content, and reserve aggressive compression for content that likely won't be needed verbatim. It can also mitigate errors by storing a parallel summary or key facts in text form.
- **Memory Storage Footprint:** Storing images instead of text may increase storage size (an image file of rendered text is larger in bytes than the text). However, it drastically **reduces runtime memory and compute** for the model. Disk storage is cheap; the bottleneck is tokens in context. If storage becomes an issue, images could be further JPEG-compressed or stored as PNGs efficiently. Additionally, old images could be archived to slower storage if not accessed for a long time.
- **OCR Decoding Errors:** The DeepSeek model generally recovers text well, but small OCR mistakes could occur (especially if text was very small or the font had artifacts). Implementation should consider a verification step. For example, when compressing, we could run the decoder on the image immediately to ensure it reproduces the text, and if not, adjust parameters (perhaps use a larger image or different font). In critical applications, one might store the original text as backup to avoid any chance of loss.
- **Multi-Modal Fine-tuning:** Training the LLM to use optical tokens may require specialized data (mixing images of text with normal text). If using DeepSeek-OCR's pretrained components, one must

ensure the final model still excels at *dialogue*. A potential pipeline is first use DeepSeek-OCR for pure OCR tasks, then **fine-tune the decoder on therapy conversation data**, while periodically feeding it some image-based memories to condition it to handle those. There's a trade-off in complexity of training versus performance: using two separate models (one for OCR, one for conversation) is simpler but less efficient; a unified model is elegant but harder to train.

- **Latency and Throughput:** Each turn involves possibly encoding images and the LLM generating output. With vLLM and GPU acceleration, the system should handle this in real-time (within a few seconds per reply). But if a user references a lot of old context at once, the system might need to retrieve and feed multiple images, which increases prompt length (though still far less than equivalent text). We must monitor that the added vision tokens don't slow generation too much. In practice, a few hundred extra tokens is minor for modern LLMs.
- **vLLM Configuration:** To make full use of the context length extension, the model's max context in vLLM should be configured to a high value (e.g. 32k or more tokens). Even though single prompts won't reach that in text, the combined vision+text tokens could. vLLM's memory paging will handle it, but the limit should be set above the expected usage. Also, using quantization (e.g. INT8) can allow a larger model or longer context to run on the same hardware [34].
- **Privacy and Security:** In therapy, data is sensitive. Storing conversation as images has an unexpected benefit: it's a bit like encryption through obscurity (not readily searchable by typical means). However, one must treat these images with the same security as text logs. Ensure the image storage is encrypted or access-controlled. Also, if using any external OCR or logging, avoid exposing these images. All processing should remain in a secure environment. Memory decay also serves privacy by not keeping detailed records forever; only high-level summaries remain for old sessions (which might be a desired feature to align with data minimization principles).
- **Fail-safe and Fallback:** The system should have fallback behavior if the optical memory fails. For instance, if the vision encoder has an error or an image is corrupted, the system might fall back to a stored text summary to at least provide some context. It's wise to keep critical facts redundantly (e.g. "important milestones" in the therapy could be kept as notes in text form in addition to being in the transcripts).
- **Extendability:** The architecture is general – it could be extended beyond couples therapy. For example, it can support group chats or other domains with long context (customer support history, personal lifelong assistants, etc.) simply by adjusting the fine-tuning. DeepSeek-OCR provides a **"JPEG for text tokens"** that could be plugged into many LLM systems [37]. This design ensures modular replacement too: if a better optical compressor comes along, we can swap out DeepSeek for it, as long as the LLM can consume that format.

In conclusion, this system design achieves a balance between **context length and computational efficiency**. By leveraging DeepSeek-OCR's optical compression, it sidesteps the scaling limits of traditional attention – allowing an LLM-based therapist to remember *entire relationship histories* (hundreds of pages of dialogue) without massive computational overhead. The combination of real-time processing (for immediate interactions) and offline maintenance (for memory optimization) ensures both responsiveness and longevity. This paves the way for more *human-like conversations*, where the AI counselor can genuinely recall past sessions and follow the couple's narrative arc over time, all while running on practical infrastructure enabled by vLLM. The design embodies the principle that sometimes, to give an AI a better memory, **it needs to forget like humans do – and *see* rather than rote-read** [6] [38]. The result is a rich, efficient, and scalable couples therapy chat system.

---

1 3 14 15 16 17 24 DeepSeek-OCR: Contexts Optical Compression
https://arxiv.org/html/2510.18234v1

2 4 28 36 DeepSeek-OCR Explained: How Contexts Optical Compression Redefines AI Efficiency
https://www.bentoml.com/blog/deepseek-ocr-contexts-optical-compression-explained

5 6 8 18 19 21 22 27 37 DeepSeek OCR is here. How to use DeepSeek OCR for free? | by Mehul Gupta | Data Science in Your Pocket | Oct, 2025 | Medium
https://medium.com/data-science-in-your-pocket/deepseek-ocr-is-here-37096b562bb0

7 12 13 20 23 Breaking LLM Long Text Processing! DeepSeek-OCR Launches Visual Memory Compression Mechanism to Solve AI Memory Bottlenecks
https://www.aibase.com/news/22115

9 25 26 38 Is DeepSeek-OCR Revealing the Blueprint for the Next Generation of AI Cognitive Memory Systems? | Small Island Research Notes
https://smallislandresearchnotes.com/deepseek-ocr-cognitive-memory-blueprint/

10 11 29 30 31 32 33 34 35 vLLM: A High-Performance Inference Engine for LLMs | by Murad Daryousse | Sep, 2025 | Medium
https://medium.com/@mdaryousse.ds/vllm-a-high-performance-inference-engine-for-llms-0b9c6c18312c