Integrating a Memory Bank into BiomedParse

# Introduction

BiomedParse is a foundation model designed for biomedical image parsing that unifies segmentation, detection, and recognition tasks. Meanwhile, MedSAM-2 treats medical images like videos, leveraging a memory mechanism to maintain coherence between successive frames. This document builds upon the initial concepts to explore how a similar memory mechanism can be integrated into BiomedParse.

# 1. BiomedParse Architecture Breakdown

BiomedParse follows a modular structure composed of the following components:

**•** Image Encoder: Extracts features from individual images or slices.

**•** Text Encoder: Interprets and processes natural language instructions.

**•** Mask Decoder: Uses extracted features to generate segmentation masks.

**•** Meta-object Classifier: Associates images with a broad ontology of possible objects and tissues.

# 2. Memory Mechanism in MedSAM-2

MedSAM-2 implements a strategy known as "memory attention." This mechanism selectively retains information from multiple slices within 2D/3D image series. The result is improved tracking of objects across slices, reducing the need for repeated prompts and ensuring continuity throughout the volume.

# 3. Proposed Memory Bank Integration Approaches

## 3.1 Stateful Encoder

The stateful encoder concept extends the image encoder to handle a stream of correlated images. Instead of processing each slice independently, it integrates a key-value recall module that stores partial embeddings from previous slices. This method allows the model to refer back to earlier computations when processing new slices.

## In-Depth Look at the Stateful Encoder

A stateful encoder surpasses the traditional stateless approach by retaining an internal memory of previous computations. This is crucial when dealing with sequential or volumetric data—like 3D image series—where maintaining spatial consistency between slices is essential.

### Fundamental Concepts:

**•** Maintaining Sequential Context: The encoder saves embeddings (feature representations) from earlier slices, which can help in interpreting the current slice in the context of what has been seen before.

**•** Using Key-Value Structures: Each slice's embedding is stored as a key-value pair in a memory bank. The key captures critical attributes (e.g., an anatomical feature), while the value stores the contextual details.

**•** Aggregating via Cross-Attention: When new slices are processed, a cross-attention module queries the memory bank to retrieve relevant historical embeddings. This integration typically results in more consistent and accurate segmentation.

### Practical Benefits:

**•** Consistency Across Slices: By "remembering" the location of key structures in previous slices, the model can produce a uniform segmentation across the entire volume.

**•** Reduced Re-Prompting: With retained contextual information, the model needs less repeated prompting, streamlining the processing of long image sequences.

**•** Enhanced Efficiency: For data types with numerous slices (e.g., MRI or CT scans), this approach minimizes errors or segmentation inconsistencies when images are processed in isolation.

### Implementation Strategies for a Stateful Encoder:

**1.** Extend the Image Encoder: Incorporate a memory component into the encoder that functions similarly to mechanisms found in transformer models. This may involve creating dedicated layers for generating and storing embeddings.

**2.** Employ Key-Value Mechanisms: As each slice is processed, generate an embedding and save it in a dynamic memory bank structured as key-value pairs.

**3.** Incorporate Cross-Attention: Add a module that queries the memory bank for relevant embeddings when a new slice is processed, ensuring that the local features are enriched with historical context.

**4.** Design Memory Update Strategies: Establish rules to update the memory bank such as using decay factors to diminish older data and prominence factors to highlight significant features.

### Practical Example:

Imagine segmenting a tumor that spans several slices of a CT scan. A traditional encoder would process each slice independently, possibly resulting in slight inconsistencies. By contrast, with a stateful encoder:

**•** The first slice is processed and its embedding—highlighting the tumor—is stored in memory.

**•** When subsequent slices are processed, the cross-attention module accesses the stored tumor-related features and combines them with the new slice's local data.

**•** The final output is a segmentation that remains consistent across all slices, even when individual slices present challenges such as lower contrast.

## 3.2 Additional Integration Approaches

**•** Cross-Attention within the Mask Decoder: Integrate memory embeddings directly into the mask decoder alongside local visual features, updating the memory with every new slice.

**•** Query Frame and Key Slice: Process an initial reference slice to extract a key embedding that can be projected onto subsequent slices via cross-attention.

**•** Update Mechanisms: Adopt a self-sorting memory bank that applies a decaying factor to older states and a prominence factor for highly visible features, similar to the approach in MedSAM-2.

**•** Alignment of Formats: Ensure that the embedding dimensions across the image encoder, memory bank, and decoder are compatible. Adjust feature map sizes if necessary to maintain consistency.

# Conclusion

Integrating a memory bank into BiomedParse represents a promising avenue for enhancing segmentation coherence and reducing the redundancy of repeated prompts. While it introduces additional computational considerations, the benefits—especially in handling sequential or volumetric data—could significantly improve performance in biomedical image parsing.