Algorithm 1 Chain-of-Merging (CoM) Framework

Require: Vision-Language Model M_{vlm} with parameters Θ_{vlm}, Large Language Model M_{llm} with parameters Θ_{llm}.
Require: Input Image I, Question Q.
Require: Hyperparameters: Penalty coefficient λ, Matching threshold τ_{match}, Weight range $[w_{min}, w_{max}]$.
Ensure: Merged Model Parameters $\Theta_{\mathcal{M}}$, Final Answer A.

// Stage 1: Initialization
Generate caption $C \leftarrow M_{vlm}(I)$
Extract hidden states $H_{vlm} \leftarrow f_{vlm}(I, Q; \Theta_{vlm})$ $\{H_{vlm} = \{h^i_{vlm}\}_{i=1}^{L}\}$
Extract hidden states $H_{llm} \leftarrow f_{llm}(C, Q; \Theta_{llm})$ $\{H_{llm} = \{h^j_{llm}\}_{j=1}^{L}\}$
// Stage 2: Adaptive Layer Matching
Initialize Merging Plan $\mathcal{P} \leftarrow \emptyset$
Target Sequence Length $L_{seq} \leftarrow \lfloor \min(\text{len}(H_{vlm}), \text{len}(H_{llm}))/2 \rfloor$
for $i = 1$ to L **do**
  $j^* \leftarrow -1$, $S_{max} \leftarrow -\infty$
  for $j = 1$ to L **do**
    {Dimensionality Reduction for CKA}
    $\tilde{h}^i_{vlm}, \tilde{h}^j_{llm} \leftarrow \text{AdaptivePool}(h^i_{vlm}, h^j_{llm}, L_{seq})$
    {Calculate Scores (Eq. 3, 6, 7)}
    $S_{struct} \leftarrow \text{CKA}(\tilde{h}^i_{vlm}, \tilde{h}^j_{llm})$
    $S_{sem} \leftarrow \text{CosineSim}(\text{MeanPool}(h^i_{vlm}), \text{MeanPool}(h^j_{llm}))$
    $S_{base} \leftarrow (S_{struct} + S_{sem})/2$
    $S^{(i,j)}_{match} \leftarrow S_{base} - \lambda \cdot |\frac{i-j}{L}|$
    if $S^{(i,j)}_{match} > S_{max}$ **then**
      $S_{max} \leftarrow S^{(i,j)}_{match}$, $j^* \leftarrow j$
    end if
  end for
  if $S_{max} > \tau_{match}$ **then**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{(i, j^*, S_{max})\}$
  end if
end for
// Stage 3: Dynamic Weight Merging
Initialize $\Theta_{\mathcal{M}} \leftarrow \Theta_{vlm}$ {Copy VLM parameters as base}
for each layer $i \in \{1, \ldots, L\}$ **do**
  if $\exists (i, j^*, S_{match}) \in \mathcal{P}$ **then**
    {Calculate Dynamic Weight (Eq. 9)}
    $S_{norm} \leftarrow (S_{match} - \tau_{match})/(1.0 - \tau_{match})$
    $w \leftarrow w_{min} + (w_{max} - w_{min}) \cdot S_{norm}$
    {Perform SLERP (Eq. 10-12)}
    $\theta^i_{merged} \leftarrow \text{SLERP}(\theta^i_{vlm}, \theta^{j^*}_{llm}, w)$
    $\Theta^i_{\mathcal{M}} \leftarrow \theta^i_{merged}$
  else
    $\Theta^i_{\mathcal{M}} \leftarrow \theta^i_{vlm}$ {Keep original VLM layer}
  end if
end for
// Inference
$A \leftarrow \text{Generate}(I, Q; \Theta_{\mathcal{M}})$
return A

A. Chain-of-Merging (CoM) Algorithm Pseudocode

In this section, we present the comprehensive algorithmic details of the proposed Chain-of-Merging (CoM) framework. As illustrated in Algorithm 1, the framework is designed to operate in a strictly training-free manner, dynamically integrating the reasoning capabilities of a Large Language Model (LLM) into a Vision-Language Model (VLM) during inference.

The execution flow is structured into three distinct phases to ensure optimal parameter integration:

1. **Initialization:** The process begins by extracting hidden representations from both models. Crucially, this stage generates a textual caption from the image to bridge the modality gap, allowing the LLM to process visual context alongside the textual question.
2. **Adaptive Layer Matching:** Unlike conventional methods that enforce rigid one-to-one mapping, our algorithm dynamically identifies the optimal layer correspondence. It computes a composite score derived from structural consistency (via CKA) and semantic similarity, penalized by layer distance to maintain architectural coherence.
3. **Dynamic Weight Merging:** Finally, the algorithm constructs the merged model parameters. Instead of simple linear averaging, we employ Spherical Linear Interpolation (SLERP) weighted by the normalized matching scores. This ensures that the geometric properties of the parameter space are preserved, minimizing the risk of feature collapse.

This algorithmic design allows CoM to adaptively adjust its merging strategy for every specific image-question pair, maximizing reasoning performance while maintaining computational feasibility.

B. Dimensionality Reduction for Efficient CKA Computation

In the Adaptive Layer Matching stage, we employ Centered Kernel Alignment (CKA) to evaluate the structural consistency between the hidden representations of the Vision-Language Model (VLM), denoted as $H_{vlm} \in \mathbb{R}^{n_{vlm} \times d}$, and the Large Language Model (LLM), denoted as $H_{llm} \in \mathbb{R}^{n_{llm} \times d}$. A fundamental challenge in this process is the discrepancy in sequence lengths: n_{vlm} includes both image and text tokens, whereas n_{llm} consists solely of caption and text tokens.

Standard CKA necessitates the computation of Gram matrices $K = H_{vlm}H^T_{vlm}$ and $L = H_{llm}H^T_{llm}$. When $n_{vlm} \neq n_{llm}$, the resulting matrices have mismatched dimensions, rendering the Hilbert-Schmidt Independence Criterion (HSIC) mathematically undefined for direct comparison. Furthermore, computing Gram matrices for high-

resolution VLMs incurs a computational complexity of $O(n^2)$, which is prohibitively expensive. To resolve these issues, we implement an **Adaptive Mean Pooling** strategy that aligns sequence lengths while preserving global topological structure.

B.1. Target Length Determination

To enable valid kernel operations, we first establish a unified target sequence length, L_{target}. This length is determined dynamically to balance the preservation of information granularity with computational efficiency. We define L_{target} based on the shorter of the two sequences, reduced by a factor of 2:

$$L_{target} = \max\left(1, \left\lfloor \frac{\min(n_{vlm}, n_{llm})}{2} \right\rfloor\right) \tag{14}$$

The inclusion of the $\max(1, \cdot)$ operation ensures numerical stability by preventing a zero target length in edge cases, while the reduction factor serves to accelerate the subsequent matrix operations.

B.2. Adaptive Pooling Logic

We apply 1D Adaptive Average Pooling along the sequence dimension to map an arbitrary input hidden state tensor $H \in \mathbb{R}^{n \times d}$ to a reduced representation $\tilde{H} \in \mathbb{R}^{L_{target} \times d}$.

For the i-th output vector in the reduced sequence (where $0 \leq i < L_{target}$), the pooling window is calculated dynamically. This approach handles non-integer stride lengths uniformly, ensuring that every token in the original sequence contributes to the pooled representation without overlap or omission gaps. The window indices are defined as:

$$\text{start\_idx} = \lfloor i \times \frac{n}{L_{target}} \rfloor \tag{15}$$

$$\text{end\_idx} = \min\left(n, \lceil (i+1) \times \frac{n}{L_{target}} \rceil\right) \tag{16}$$

The pooled feature vector $\tilde{H}[i]$ is computed as the mean of the hidden states within this window:

$$\tilde{H}[i] = \frac{1}{\text{end\_idx} - \text{start\_idx}} \sum_{k=\text{start\_idx}}^{\text{end\_idx}-1} H[k] \tag{17}$$

B.3. HSIC Computation

Following the adaptive pooling, both the reduced VLM tensor $\tilde{H}_{vlm}$ and LLM tensor $\tilde{H}_{llm}$ share the dimensions $L_{target} \times d$. We subsequently compute the aligned Gram matrices $K = \tilde{H}_{vlm}\tilde{H}_{vlm}^T$ and $L = \tilde{H}_{llm}\tilde{H}_{llm}^T$, both of size $L_{target} \times L_{target}$. This alignment allows for the efficient computation of the HSIC value:

$$\text{HSIC}(K, L) = \frac{1}{(L_{target} - 1)^2} \text{tr}(KHLH) \tag{18}$$

By focusing on the global topological correspondence of the feature spaces rather than strict token-to-token alignment, this method effectively bridges the modality gap between visual and textual representations.