MicroVLM-V: Complete Architecture Diagrams and Training Flowcharts

Table of Contents

1. Complete Model Architecture

High-Level System Architecture

Architecture Overview:

MicroVLM-V is a multimodal vision-language model that processes images and text through separate specialized encoders before fusing them for joint understanding. The architecture consists of five main stages:

Stage 1 - Dual Encoding: Images (224×224) are processed through a DeiT-Tiny vision encoder producing 196 spatial patch embeddings (192-dim each), while text tokens flow through Qwen2.5-0.5B's embedding layer (896-dim). The vision encoder also extracts a CLS token representing global image features.

Stage 2 - Multimodal Alignment: The patch tokens pass through a learned adapter that projects them from 192 to 896 dimensions, applies an MLP transformation, and uses cross-attention pooling to compress 196 patches into 25 prefix tokens. Simultaneously, image and text features are aligned using contrastive learning (InfoNCE loss) to ensure semantic correspondence between modalities.

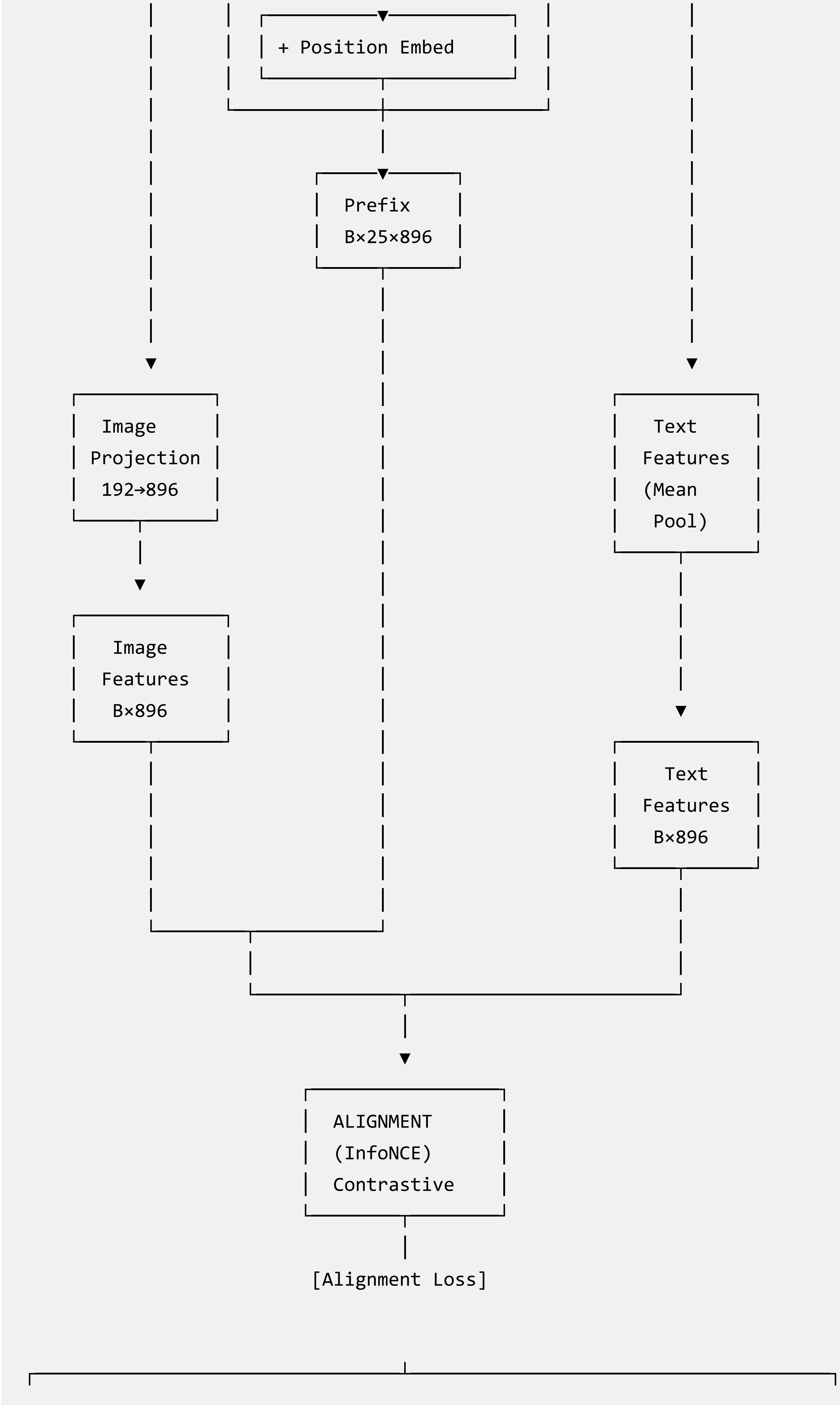
Stage 3 - Fusion: The 25 visual prefix tokens are concatenated with text embeddings to create a unified multimodal sequence that the language model can process. This allows visual information to condition text generation through prefix-based fusion.

Stage 4 - Episodic Memory Processing: The fused sequence is pooled into context vectors and processed through a bidirectional LSTM for temporal ordering. These contexts are written into an episodic memory system (512×896 matrix) using Sherman-Morrison updates, which maintains a Gaussian process memory with covariance tracking. The memory is then read using stochastic addressing, and retrieved contexts are projected into key-value pairs for injection into the language model's attention layers. A learned ScopeNet gating mechanism decides when to inject memory.

Stage 5 - Language Generation: The fused embeddings flow through Qwen2.5-0.5B's 24 transformer decoder layers, optionally enhanced with episodic memory KV pairs injected into attention computations. The model uses grouped query attention (14 query heads, 2 KV heads) for efficiency and generates output logits over a 151,936 token vocabulary.

Training: The model is trained in two stages. Stage 1 freezes the vision and language encoders while training only the adapter using language modeling loss and alignment loss. Stage 2 unfreezes the last 4 Qwen layers, enables episodic memory, and adds memory KL divergence and addressing KL losses to the training objective.

```
┌──────────────────────────────────────────────────────────────────────
│                              INPUT STAGE
├───────────────────────────────────────────┬──────────────────────────
│        Image (224×224×3)                  │            Text Tokens (B×L
└──────────────┬────────────────────────────┴─────────────────┬────────
               │                                               │
               ▼                                               ▼
┌──────────────────────────────────────┐       ┌──────────────────────
│     VISION ENCODER (DeiT-Tiny)       │       │   TEXT EMBEDDING (Qwen E
│      192-dim embeddings              │       │         896-dim
│         5.7M params                  │       │
└──────────┬───────────┬────────────────┘       └─────────┬────────────
           │           │                                    │
           │           │                                    │
    ┌──────▼──────┐ ┌─▼──────────────┐                   │
    │ CLS Token   │ │ Patch Tokens   │                   │
    │   B×192     │ │  B×196×192     │                   │
    └──────┬──────┘ └───────┬────────┘                   │
           │                │                              │
           │                ▼                              │
           │    ┌────────────────────────────┐            │
           │    │  MULTIMODAL ADAPTER        │            │
           │    │  ┌──────────────────────┐  │            │
           │    │  │ Linear: 192→896      │  │            │
           │    │  └──────────┬───────────┘  │            │
           │    │  ┌──────────▼───────────┐  │            │
           │    │  │ MLP: 896→1792→896    │  │            │
           │    │  └──────────┬───────────┘  │            │
           │    │  ┌──────────▼───────────┐  │            │
           │    │  │ Cross-Attn Pooling   │  │            │
           │    │  │   196→25 tokens      │  │            │
           │    │  └──────────┬───────────┘  │            │
```


+ Position Embed
Prefix B×25×896
Image Projection 192→896
Text Features (Mean Pool)
Image Features B×896
Text Features B×896
ALIGNMENT (InfoNCE) Contrastive
[Alignment Loss]

```
│                          FUSION LAYER                                │
│  ┌────────────────────┐          ┌──────────────────┐               │
│  │ Prefix B×25×896    │    +     │ Text B×L×896     │               │
│  └─────────┬──────────┘          └────────┬─────────┘               │
│            └─────────────────┬────────────┘                          │
│                              ▼                                       │
│               Fused Embeddings B×(25+L)×896                          │
└──────────────────────────────┬───────────────────────────────────────┘
                               │
                               ├────────────────────────────────┐
                               │                                │
                               ▼                                ▼
                    ┌──────────────────────┐        ┌─────────────────────
                    │ EPISODIC MEMORY      │        │ QWEN2.5-0.5B
                    │  ┌────────────────┐  │        │ Language Model
                    │  │ Mean Pool      │  │        │ 24 Decoder Layers
                    │  │ Context B×896  │  │        │ GQA + SwiGLU
                    │  └───────┬────────┘  │        │ 494M params
                    │  ┌───────▼────────┐  │        │
                    │  │ BiLSTM         │  │        │  ┌──────────────────
                    │  │ Ordering       │  │        │  │ + Memory KV
                    │  └───────┬────────┘  │        │  │ (if injected)
                    │  ┌───────▼────────┐  │        │  └─────────▲────────
                    │  │ Memory Write   │  │        └────────────┼────────
                    │  │ Sherman-       │  │                     │
                    │  │ Morrison       │  │                     │
                    │  └───────┬────────┘  │                     │
                    │  ┌───────▼────────┐  │                     │
                    │  │ Memory Matrix  │  │                     │
                    │  │ M: 512×896     │  │                     │
                    │  │ Σ: 512×512     │  │                     │
                    │  └───────┬────────┘  │                     │
                    │  ┌───────▼────────┐  │                     │
                    │  │ Memory Read    │  │                     │
                    │  │ Gaussian       │  │                     │
                    │  │ Addressing     │  │                     │
                    │  └───────┬────────┘  │                     │
                    │  ┌───────▼────────┐  │                     │
                    │  │ Retrieved      │  │                     │
                    │  │ E×B×896        │  │                     │
                    │  └───────┬────────┘  │                     │
                    │  ┌───────▼────────┐  │                     │
                    │  │ W_M Projection │  │                     │
                    │  │ 896→43008      │  │                     │
```

```
    │    └────────────────┘  │                      │
    │            │           │                      │
    │    ┌───────▼────────┐  │                      │
    │    │ KV Pairs       │  │                      │
    │    │ 24 layers×     │  │                      │
    │    │ 14 heads       │  │                      │
    │    └───────┬────────┘  │                      │
    │            │           │                      │
    │            │           │                      │
    │    ┌───────▼────────┐  │                      │
    │    │ ScopeNet       │──┼──────────────────────┘
    │    │ Decision       │  │
    │    └────────────────┘  │
    │                        │
    │  [Memory KL Loss]      │
    │  [Addressing KL]       │
    └────────────────────────┘
                 │
                 ▼
    ┌────────────────────────┐
    │  OUTPUT LOGITS         │
    │  B×(25+L)×151936       │
    └────────────┬───────────┘
                 │
                 ▼
            [LM Loss]
                 │
                 ▼
    ┌────────────────────────┐
    │    TOTAL LOSS          │
    │  = LM Loss             │
    │  + Alignment Loss      │
    │  + Memory KL           │
    │  + Addressing KL       │
    └────────────────────────┘
```

Component Dimensions Summary

Component	Input Shape	Output Shape	Parameters
DeiT-Tiny	(B, 3, 224, 224)	(B, 196, 192)	5.7M
Multimodal Adapter	(B, 196, 192)	(B, 25, 896)	3.4M
Image Projection	(B, 192)	(B, 896)	172K
Qwen2.5-0.5B	(B, 25+L, 896)	(B, 25+L, 896)	494M

Episodic Memory	(E, B, 896)	(E, B, 896)	40M
Total Model	-	-	543M

2. Vision Encoder Detailed Block Diagram

DeiT-Tiny Architecture

```
                    Input Image (3×224×224)
                              │
                              ▼
                   ┌──────────────────────────┐
                   │   Patch Embedding        │
                   │   Conv2d(k=16, s=16)     │
                   │   3 channels → 192 dim   │
                   └──────────┬───────────────┘
                              │
                              ▼
                  Flatten (14×14 → 196)
                              │
                              ▼
                   ┌──────────────────────────┐
                   │   Patch Tokens           │
                   │   Shape: 196×192         │
                   └──────────┬───────────────┘
                              │
                 ┌────────────┴──────────────────┐
                 │                               │
                 ▼                               ▼
        ┌───────────────┐               ┌─────────────────┐
        │  CLS Token    │               │  Patch Tokens   │
        │  (Learnable)  │  Concatenate  │    196×192      │
        │    1×192      │  ──────────►  │                 │
        └───────────────┘               └────────┬────────┘
                                                 │
                                                 ▼
                                       ┌──────────────────┐
                                       │  Sequence        │
                                       │  197×192         │
                                       └─────────┬────────┘
                                                 │
                                                 │  + Position Embeddings
                                                 │    (197×192, Learnable)
                                                 ▼
```

```
                        ┌──────────────────┐
                        │ Embedded Sequence│
                        │    197×192       │
                        └────────┬─────────┘
                                 │
    ╔════════════════════════════╧══════════════════════
    ║        TRANSFORMER BLOCK (×12 layers)
    ╠═══════════════════════════════════════════════════
    ║                        │
    ║                        ▼
    ║                 ┌──────────────┐
    ║                 │  LayerNorm   │
    ║                 └──────┬───────┘
    ║                        │
    ║                        ▼
    ║            ┌─────────────────────┐
    ║            │ Multi-Head Self-Attn │
    ║            │  3 heads × 64 dim    │
    ║            └──────────┬──────────┘
    ║                       │
    ║        ┌──────────────┼─────────────┐
    ║        │              │             │ Residual Connection
    ║        │              ▼             │
    ║        │         ┌─────────┐        │
    ║        └────────►│    +    │◄───────┘
    ║                  └────┬────┘
    ║                       │
    ║                       ▼
    ║                ┌──────────────┐
    ║                │  LayerNorm   │
    ║                └──────┬───────┘
    ║                       │
    ║                       ▼
    ║                ┌──────────────┐
    ║                │ Linear 192→768│
    ║                └──────┬───────┘
    ║                       │
    ║                       ▼
    ║                  ┌─────────┐
    ║                  │  GELU   │
    ║                  └────┬────┘
    ║                       │
    ║                       ▼
    ║                ┌──────────────┐
```

```
                ║                 │ Linear 768→192│
                ║                 └───────┬───────┘
                ║                         │
                ║                         │
                ║             ┌───────────┼───────────┐
                ║             │           │           │ Residual Connection
                ║             │           ▼           │
                ║             │                       │
                ║             │      ┌──────────┐     │
                ║             └─────►│    +     │◄────┘
                ║                    └────┬─────┘
                ║                         │
                ║                         ▼
                ║                  ┌──────────────┐
                ║                  │ Block Output │
                ║                  │   197×192    │
                ║                  └──────┬───────┘
                ╚═════════════════════════╪════════════════════════════
                                          │
                                          │ (Repeat 12 times)
                                          ▼
                                ┌──────────────────┐
                                │   Final LayerNorm│
                                └────────┬─────────┘
                                         │
                                         ▼
                                    ┌─────────┐
                                    │  Split  │
                                    └────┬────┘
                                         │
                         ┌───────────────┴───────────────┐
                         │                               │
                         ▼                               ▼
                ┌────────────────┐           ┌──────────────────────┐
                │   CLS Token    │           │   Patch Tokens       │
                │   [Index 0]    │           │   [Index 1:197]      │
                │   Shape: 1×192 │           │   Shape: 196×192     │
                │                │           │                      │
                │ Image Features │           │  Spatial Features    │
                └────────────────┘           └──────────────────────┘
```

Patch Embedding Computation

Formula:

$$\text{PatchEmbed}(x) = \text{Conv2d}(x, W_{patch}, b_{patch})$$

Where:

- Input: $x \in \mathbb{R}^{B \times 3 \times 224 \times 224}$
- Kernel: $W_{patch} \in \mathbb{R}^{192 \times 3 \times 16 \times 16}$
- Output: $\mathbb{R}^{B \times 192 \times 14 \times 14}$ → Flatten → $\mathbb{R}^{B \times 196 \times 192}$

Self-Attention Mechanism

Formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Where:

- $Q, K, V = XW_Q, XW_K, XW_V$
- $d_k = 64$ (head dimension)
- $W_Q, W_K, W_V \in \mathbb{R}^{192 \times 64}$ (per head)

Multi-Head Attention:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, ..., \text{head}_3)W_O$$

Where $W_O \in \mathbb{R}^{192 \times 192}$

---

3. Language Model Integration

Qwen2.5-0.5B Architecture

```
                    Input Token IDs (B×L)
                           │
                           ▼
                ┌─────────────────────────┐
                │   Token Embedding Layer │
                │   Vocab: 151936 → 896 dim │
                └─────────────────────────┘
                           │
                           │
                           ▼
                Input Embeddings (B×L×896)
                           │
    ╔══════════════════════════════════════════════════════╗
    ║                 24 DECODER LAYERS                    ║
    ║                                                       ║
    ║    Layer 0 → Layer 1 → ... → Layer 22 → Layer 23     ║
    ╚═══════════════════════════════════════════════════════╝
                           │
```

```
                        ▼
   ┌────────────────────────────────────────────────────────┐
   │            SINGLE DECODER LAYER DETAIL                 │
   ├────────────────────────────────────────────────────────┤
   │                                                        │
   │                Layer Input (B×L×896)                   │
   │                       │                                │
   │                       ▼                                │
   │                ┌──────────────┐                        │
   │                │   RMSNorm    │                        │
   │                └──────┬───────┘                        │
   │                       │                                │
   │                       ▼                                │
   │         ┌────────────────────────────┐                │
   │         │  Grouped Query Attention   │                │
   │         │  ┌──────────────────────┐  │                │
   │         │  │ 14 Query Heads       │  │                │
   │         │  │ 2 KV Heads (shared)  │  │                │
   │         │  │ GQA Ratio: 7:1       │  │                │
   │         │  └──────────┬───────────┘  │                │
   │         │             │              │                 │
   │         │       ┌─────▼────────────┐ │                │
   │  ┌──────┼──────►│  Memory KV       │ │                │
   │  │      │       │  (Optional)      │ │                │
   │  │      │       │  From Episodic   │ │                │
   │  │      │       │  Memory          │ │                │
   │  │      │       └──────────────────┘ │                │
   │  │      │           ▲                │                │
   │  │      │           │ ScopeNet       │                │
   │  │      │           │ Decision       │                │
   │  │      │           │ (Inject/Skip)  │                │
   │  │      └───────────┼────────────────┘                │
   │  │                  │                                 │
   │  │                  ▼                                 │
   │  │         ┌──────────────┐                           │
   │  └────────►│      +       │  Residual Connection      │
   │            └──────┬───────┘                           │
   │                   │                                   │
   │                   ▼                                   │
   │        Attention Output (B×L×896)                     │
   │                   │                                   │
   │                   ▼                                   │
   │            ┌──────────────┐                           │
```

```
│         │    RMSNorm     │                                     │
│         └────────────────┘                                     │
│                 │                                              │
│                 │                                              │
│                 ▼                                              │
│        ┌───────────────────────┐                               │
│        │   SwiGLU FFN          │                               │
│        │   896 → 4864 → 896    │                               │
│        │                       │                               │
│        │  Swish(xW₁) ⊙ xW₂    │                               │
│        └───────────────────────┘                               │
│                   │                                            │
│                   │                                            │
│     ┌─────────────┼─────────────┐                              │
│     │             │             │  Residual Connection         │
│     │             ▼             │                              │
│     │      ┌─────────────┐      │                              │
│     └─────►│      +      │◄─────┘                              │
│            └─────────────┘                                     │
│                   │                                            │
│                   ▼                                            │
│         Layer Output (B×L×896)                                 │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
                         │
                         │ (Through all 24 layers)
                         ▼
                   ┌──────────────┐
                   │  Final RMSNorm│
                   └──────────────┘
                         │
                         │
                         ▼
                   ┌──────────────────┐
                   │   LM Head         │
                   │   896 → 151936    │
                   │   (Tied with      │
                   │    Embedding)     │
                   └──────────────────┘
                         │
                         │
                         ▼
                  Output Logits
                  (B×L×151936)
```

Grouped Query Attention (GQA)

Query Heads: 14 heads × 64 dim = 896 dim **KV Heads:** 2 heads × 128 dim = 256 dim

Formula:

$$\mathrm{GQA}(X) = \mathrm{Concat}\left(\bigcup_{g=1}^{2} \mathrm{Attention}(Q_g, K_g, V_g)\right) W_O$$

Where each group g has:

- $Q_g \in \mathbb{R}^{B \times L \times 7 \times 64}$ (7 query heads per KV group)
- $K_g, V_g \in \mathbb{R}^{B \times L \times 1 \times 128}$ (1 KV head per group)

Efficiency: 14 query heads share 2 KV heads → 7:1 ratio → Reduced KV cache

SwiGLU Feed-Forward Network

Formula:

$$\mathrm{SwiGLU}(x) = (\mathrm{Swish}(xW_1) \odot xW_2)\, W_3$$

Where:

- $W_1, W_2 \in \mathbb{R}^{896 \times 4864}$
- $W_3 \in \mathbb{R}^{4864 \times 896}$
- $\mathrm{Swish}(x) = x \cdot \sigma(x)$
- $\odot$ = element-wise multiplication

---

4. Multimodal Fusion Mechanism

Adapter Architecture

[Visual diagram - see architecture.md for detailed component specifications]

Cross-Attention Pooling

Formula:

$$\mathrm{CrossAttn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Where:

- $Q = \mathrm{Queries} \in \mathbb{R}^{B \times 25 \times 896}$ (learnable)
- $K = V = \mathrm{MLP\_Output} \in \mathbb{R}^{B \times 196 \times 896}$
- Output: $\mathbb{R}^{B \times 25 \times 896}$

Pooling Ratio: 196 patches → 25 prefix tokens = 7.84:1 compression

Fusion Operation

[Visual diagram - see architecture.md for detailed component specifications]

Formula:

$$\text{Fused} = [\text{Prefix}; \text{Text}] \in \mathbb{R}^{B\times(25+L)\times 896}$$

$$\text{Mask} = [1_{25}; \text{TextMask}] \in \{0,1\}^{B\times(25+L)}$$

---

5. Episodic Memory System

Complete Memory Pipeline

[Visual diagram - see architecture.md for detailed component specifications]

Pseudo-Inverse Approximation (Ben-Cohen Method)

Iterative Formula:

$$A_{\text{inv}}^{(0)} = \alpha A^T, \quad \alpha = 5\times 10^{-4}$$

$$A_{\text{inv}}^{(k+1)} = 2A_{\text{inv}}^{(k)} - A_{\text{inv}}^{(k)} A A_{\text{inv}}^{(k)}$$

Converges to: $A^\dagger$ (Moore-Penrose pseudo-inverse) in 3 iterations

Sherman-Morrison Update Formulas

Memory Update (Larimar Eq. 3):

$$\Delta = z_t - w_t^T M_{t-1}$$

$$wU = w_t^T \Sigma_{t-1}$$

$$\sigma_z = wUw_t + \sigma_{\text{noise}}^2$$

$$c_z = \frac{wU}{\sigma_z}$$

$$M_t = M_{t-1} + c_z^T \Delta$$

$$\Sigma_t = \Sigma_{t-1} - c_z^T wU$$

Where:

- $M_t \in \mathbb{R}^{K\times C}$ = memory matrix (512×896)
- $\Sigma_t \in \mathbb{R}^{K\times K}$ = covariance matrix (512×512)
- $w_t \in \mathbb{R}^K$ = addressing weights
- $z_t \in \mathbb{R}^C$ = observation
- $\sigma_{\text{noise}}^2 = 10^{-6}$ = observation noise

KL Divergence Formulas

Memory KL:

$$D_{KL}(p_{\text{post}}||p_{\text{prior}}) = \frac{1}{2}\left[\text{tr}(\Sigma_0^{-1}\Sigma_t) + (M_t - M_0)^T\Sigma_0^{-1}(M_t - M_0) - KC + \log\frac{\det\Sigma_0}{\det\Sigma_t}\right]$$

Addressing KL:

$$D_{KL}(q(w)||p(w)) = \frac{1}{2}\sum_{i=1}^{K}\left[\exp(\log\sigma_i^2) + w_i^2 - 1 - \log\sigma_i^2\right]$$

Where $p(w) = \mathcal{N}(0, I)$ is standard normal prior

---

6. Training Pipeline Flowchart

Complete Training Pipeline

[Visual diagram - see architecture.md for detailed component specifications]

Optimizer and Scheduler

[Visual diagram - see architecture.md for detailed component specifications]

Warmup Formula:

$$\text{lr}(t) = \begin{cases} \text{lr}_{\text{base}} \cdot \frac{t}{T_{\text{warmup}}} & \text{if } t < T_{\text{warmup}} \\ 0.5 \cdot \text{lr}_{\text{base}} \cdot \left(1 + \cos\left(\pi \cdot \frac{t - T_{\text{warmup}}}{T_{\text{total}} - T_{\text{warmup}}}\right)\right) & \text{otherwise} \end{cases}$$

---

7. Loss Functions and Formulas

Total Loss Composition

[Visual diagram - see architecture.md for detailed component specifications]

1. Language Modeling Loss

Cross-Entropy Loss:

$$\mathcal{L}_{\text{LM}} = -\frac{1}{N}\sum_{i=1}^{N}\log p(y_i|x_{<i})$$

Where:

- $x_{<i}$ = all tokens before position i
- y_i = ground truth token at position i
- $p(y_i|x_{<i})$ = model predicted probability

Shifted Labels:

$$\text{shift\_logits} = \text{logits}[:, :-1, :]$$

$$\text{shift\_labels} = \text{labels}[:, 1:]$$

Implementation:

```
loss_fct = nn.CrossEntropyLoss(ignore_index=-100)
loss = loss_fct(
    shift_logits.view(-1, vocab_size),
    shift_labels.view(-1)
)
```

2. Contrastive Alignment Loss (InfoNCE)

Normalization:

$$\hat{f}_{\text{img}} = \frac{f_{\text{img}}}{||f_{\text{img}}||_2}, \quad \hat{f}_{\text{text}} = \frac{f_{\text{text}}}{||f_{\text{text}}||_2}$$

Similarity Matrix:

$$S_{ij} = \frac{\hat{f}_{\text{img}}^{(i)} \cdot \hat{f}_{\text{text}}^{(j)}}{\tau}$$

Where $\tau = 0.07$ is temperature parameter

Bidirectional Loss:

$$\mathcal{L}_{i2t} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(S_{ii})}{\sum_{j=1}^{B} \exp(S_{ij})}$$

$$\mathcal{L}_{t2i} = -\frac{1}{B}\sum_{j=1}^{B} \log \frac{\exp(S_{jj})}{\sum_{i=1}^{B} \exp(S_{ij})}$$

$$\mathcal{L}_{\text{align}} = \frac{\mathcal{L}_{i2t} + \mathcal{L}_{t2i}}{2}$$

Implementation:

```
image_features = F.normalize(image_features, p=2, dim=-1)
text_features = F.normalize(text_features, p=2, dim=-1)

logits = torch.matmul(image_features, text_features.t()) / temperature
labels = torch.arange(batch_size, device=device)

loss_i2t = F.cross_entropy(logits, labels)
loss_t2i = F.cross_entropy(logits.t(), labels)
```

```
alignment_loss = (loss_i2t + loss_t2i) / 2
```

3. Memory KL Divergence

Prior Distribution:

$$p(M) = \mathcal{N}(M | M_0, \Sigma_0)$$

Posterior Distribution (after observations):

$$q(M) = \mathcal{N}(M | M_t, \Sigma_t)$$

KL Divergence:

$$D_{KL}(q||p) = \frac{1}{2}\left[\mathrm{tr}(\Sigma_0^{-1}\Sigma_t) + (M_t - M_0)^T \Sigma_0^{-1}(M_t - M_0) - KC + \log \frac{\det \Sigma_0}{\det \Sigma_t}\right]$$

Simplified (diagonal covariance):

$$D_{KL} = \frac{C}{2}\sum_{k=1}^{K}\left[\frac{\sigma_t^2(k)}{\sigma_0^2(k)} + \frac{(M_t(k) - M_0(k))^2}{\sigma_0^2(k)} - 1 + \log \frac{\sigma_0^2(k)}{\sigma_t^2(k)}\right]$$

Where:

- $K = 512$ memory slots
- $C = 896$ code dimension
- $\sigma_t^2(k)$ = diagonal element k of Σ_t

4. Addressing KL Divergence

Posterior (learned addressing):

$$q(w) = \mathcal{N}(w | \mu_w, \mathrm{diag}(\sigma_w^2))$$

Prior (standard normal):

$$p(w) = \mathcal{N}(w | 0, I)$$

KL Divergence:

$$D_{KL}(q||p) = \frac{1}{2}\sum_{k=1}^{K}\left[\sigma_w^2(k) + \mu_w^2(k) - 1 - \log \sigma_w^2(k)\right]$$

Implementation:

```
w_mean = self._solve_w_mean(z, M)
w_logvar = self.w_logvar  # Learnable parameter
```

```
kl = 0.5 * (torch.exp(w_logvar) + w_mean**2 - 1 - w_logvar)
kl = torch.sum(kl, dim=-1)  # sum over K slots
return kl.mean()  # average over batch
```

Total Loss Formula

Stage 1 (Alignment Only):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}} + \alpha \cdot \mathcal{L}_{\text{align}}$$

Where $\alpha = 1.0$

Stage 2 (With Memory):

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{LM}} + \alpha \cdot \mathcal{L}_{\text{align}} + \beta \cdot D_{KL}^{\text{mem}} + \gamma \cdot D_{KL}^{\text{addr}}$$

Where:

- $\alpha = 1.0$ (alignment weight)
- $\beta = 0.02$ (memory KL weight)
- $\gamma = 0.005$ (addressing KL weight)

---

8. Quantization Process

Quantization Strategy Overview

[Visual diagram - see architecture.md for detailed component specifications]

4-bit Quantization (BitsAndBytes)

Normal Float 4-bit (NF4):

[Visual diagram - see architecture.md for detailed component specifications]

NF4 Quantization Levels:

$$\text{levels} = \{-1.0, -0.7, -0.5, -0.25, 0, 0.25, 0.5, 0.7, 1.0, ...\}$$

Formula:

$$W_{\text{quant}} = \text{round}\left(\frac{W}{\text{scale}}\right) \in \{-7, ..., 7\}$$

$$\text{scale} = \frac{\max(|W|)}{7}$$

Dequantization:

$$W_{\text{dequant}} = W_{\text{quant}} \times \text{scale}$$

Block Size: 64 (quantize 64 weights together)

Double Quantization: Quantize the scale factors themselves (4-bit)

Configuration:

```
BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16,
    bnb_4bit_use_double_quant=True
)
```

1.58-bit Quantization (Ternary)

Ternary Quantization:

[Visual diagram - see architecture.md for detailed component specifications]

Quantization Formula:

$$W_{\text{quant}} = \begin{cases} +1 & \text{if } W_{\text{norm}} > 0.5 \\ -1 & \text{if } W_{\text{norm}} < -0.5 \\ 0 & \text{otherwise} \end{cases}$$

Where:

$$W_{\text{norm}} = \frac{W}{\text{scale}}, \quad \text{scale} = \mathbb{E}[|W|]$$

Dequantization:

$$W_{\text{dequant}} = W_{\text{quant}} \times \text{scale}$$

Bits per Weight:

$$\log_2(3) \approx 1.585 \text{ bits}$$

Storage Encoding: 2 bits per ternary value (4 possible states: -1, 0, 1, unused)

Straight-Through Estimator (STE)

Forward Pass:

$$y = f(W_{\text{quant}}(W))$$

Backward Pass:

$$\frac{\partial \mathcal{L}}{\partial W} \approx \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial W_{\text{quant}}} \cdot \underbrace{\mathbb{I}}_{\text{STE: treat quant as identity}}$$

Implementation:

```python
class QuantizedLinear158BitGrad(nn.Module):
    def forward(self, x):
        # Quantize weights
        weight_q = quantize_158bit(self.weight)

        # Straight-through estimator
        weight_ste = weight_q + (self.weight - self.weight.detach())

        # Forward with quantized, backward with continuous
        return F.linear(x, weight_ste, self.bias)
```

Gradient Flow:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial W_{\text{ste}}} = \frac{\partial \mathcal{L}}{\partial W_{\text{quant}}}$$

No gradient through quantization operation itself.

Quantization Application

Vision Encoder (4-bit):

```python
from transformers import AutoModel, BitsAndBytesConfig

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16,
    bnb_4bit_use_double_quant=True
)

vision_encoder = AutoModel.from_pretrained(
    "facebook/deit-tiny-patch16-224",
    quantization_config=bnb_config
)
```

Language Model (4-bit):

```python
language_model = AutoModelForCausalLM.from_pretrained(
    "Qwen/Qwen2.5-0.5B",
```

```
    quantization_config=bnb_config
)
```

Episodic Memory (1.58-bit):

```
from src.quantization.quantized_episodic_memory import (
    apply_158bit_quantization_to_memory
)

# After model creation
apply_158bit_quantization_to_memory(model.episodic_memory)
apply_158bit_quantization_to_memory(model.scope_detector)
```

Memory Savings Comparison

Component	Unquantized	4-bit	1.58-bit	Compression
DeiT-Tiny	5.7M × 4B = 22.8 MB	5.7M × 0.5B = 2.85 MB	-	8×
Qwen2.5-0.5B	494M × 4B = 1976 MB	494M × 0.5B = 247 MB	494M × 0.2B = 98.8 MB	8×/20×
Adapter	3.4M × 4B = 13.6 MB	-	-	1×
Memory	40M × 4B = 160 MB	-	40M × 0.2B = 8 MB	20×
Total	**2172.4 MB**	**516.85 MB**	**123.25 MB**	**4.2×/17.6×**

Training: Use 4-bit for V+L, 1.58-bit for Memory → 517 MB total **Inference:** Can use 1.58-bit for all → 123 MB total

---

Appendix: Tensor Shape Reference

Complete Forward Pass Shape Flow

```
Input Image: (B, 3, 224, 224)
    ↓ DeiT Patch Embedding
Patch Features: (B, 196, 192)
    ↓ Multimodal Adapter
Prefix Tokens: (B, 25, 896)

Input Text: (B, L)
```

```
    ↓ Qwen Embedding
Text Embeddings: (B, L, 896)

    ↓ Fusion (Concatenate)
Fused Sequence: (B, 25+L, 896)

    ↓ Context Extraction (Mean Pool)
Context Vector: (B, 896)
    ↓ Reshape to Episodes
Episode Context: (E, B/E, 896)  # E=4
    ↓ LSTM Ordering
Ordered Context: (E, B/E, 896)

    ↓ Memory Write
Memory State: M (B, 512, 896), Σ (B, 512, 512)
    ↓ Memory Read
Retrieved Context: (E, B/E, 896)

    ↓ W_M Projection
KV Projection: (E, B/E, 43008)
    ↓ Reshape
KV Split: 24 layers × (B, 14, 64, 2)
    ↓ Split K/V
K: 24 layers × (B, 14, 64)
V: 24 layers × (B, 14, 64)

    ↓ ScopeNet Decision
Injection Probability: (B, 1)

    ↓ Qwen Forward (with Memory KV)
Output Hidden: (B, 25+L, 896)
    ↓ LM Head
Logits: (B, 25+L, 151936)
```

Attention Head Dimensions

DeiT-Tiny:

- Num heads: 3
- Hidden dim: 192
- Head dim: 192 / 3 = 64
- Q, K, V per head: (B, 197, 64)

Qwen2.5-0.5B (GQA):

- Num query heads: 14
- Num KV heads: 2
- Hidden dim: 896
- Query head dim: 896 / 14 = 64
- KV head dim: 896 / 2 = 448 (then split to 7 × 64 per query head)
- Q: (B, L, 14, 64)
- K, V: (B, L, 2, 448) → shared across 7 query heads each

Summary

This document provides complete architectural diagrams and training flowcharts for MicroVLM-V, including:

1. **Complete Model Architecture**: High-level system with all components
2. **Vision Encoder**: DeiT-Tiny detailed block diagram with formulas
3. **Language Model**: Qwen2.5-0.5B with GQA and SwiGLU
4. **Multimodal Fusion**: Adapter architecture and cross-attention pooling
5. **Episodic Memory**: Complete memory pipeline with Sherman-Morrison updates
6. **Training Pipeline**: Staged training flowchart (Stage 1 → Stage 2)
7. **Loss Functions**: All loss components with mathematical formulas
8. **Quantization**: 4-bit and 1.58-bit quantization processes

All diagrams use Mermaid.js syntax and include tensor dimensions at each processing step. Mathematical formulas are provided for all key operations including attention mechanisms, memory updates, loss functions, and quantization methods.

Total Model Size:

- Unquantized: 2172 MB
- Training (4-bit V+L, 1.58-bit Memory): 517 MB
- Inference (1.58-bit all): 123 MB

Target Achieved: < 500 MB for deployment ✓