Episodic Memory–Refined Prototypical Few-Shot Learning

1. Problem Setting

We consider the standard N-way K-shot few-shot classification setting. Let $\mathcal{D}*{\text{base}}$ and $\mathcal{D}*{\text{novel}}$ denote disjoint label sets used for meta-training and meta-testing, respectively. Meta-training proceeds over a sequence of episodes ${\mathcal{E}*t}*{t=1}^T$. Each episode $\mathcal{E}_t = (\mathcal{S}_t, \mathcal{Q}_t)$ consists of a support set

$$\mathcal{S}_t = \{(x_{t,i}, y_{t,i})\}_{i=1}^{NK}$$

and a query set $\mathcal{Q}*t = {(x*{t,j}, y_{t,j})}_{j=1}^{NQ}$.

Let $f_\theta: \mathcal{X} \to \mathbb{R}^d$ be a feature encoder parameterized by θ.

2. Episodic Prototypical Representation

Within episode t, we compute class prototypes in the standard prototypical network manner:

$$\mathbf{p}_{t,c}^{(0)} = \frac{1}{|\mathcal{S}_{t,c}|} \sum_{(x,y)\in\mathcal{S}_{t,c}} f_\theta(x),$$

where $\mathcal{S}_{t,c}$ denotes the subset of support samples belonging to class c.

Unlike conventional episodic training, we do not treat episodes as independent. Instead, we introduce an *episodic memory* that accumulates and refines prototype representations across episodes.

3. Episodic Memory Module

We maintain a memory bank $\mathcal{M}*t* = *{ \mathbf{m}*{t,c} }*{c \in \mathcal{D}*{\text{base}}}$, where $\mathbf{m}_{t,c} \in \mathbb{R}^d$ stores a running meta-prototype for class c up to episode t.

3.1 Memory Initialization

At $t=0$, memory entries are initialized as empty or zero vectors. When a class c is observed for the first time, we set:

$$\mathbf{m}_{t,c} \leftarrow \mathbf{p}_{t,c}^{(0)}.$$

3.2 Prototype Refinement via Memory

For an episode t and class c, we refine the episode-specific prototype by conditioning on the memory:

$$\mathbf{p}_{t,c} = \phi\big(\mathbf{p}_{t,c}^{(0)}, \mathbf{m}_{t-1,c}\big),$$

where $\phi(\cdot)$ is a refinement function. In our implementation, we use a gated residual update:

$$\mathbf{p}_{t,c} = \alpha_{t,c}\, \mathbf{p}_{t,c}^{(0)} + (1 - \alpha_{t,c})\, \mathbf{m}_{t-1,c},$$

with

$$\alpha_{t,c} = \sigma\big(g([\mathbf{p}_{t,c}^{(0)}; \mathbf{m}_{t-1,c}])\big),$$

where $g(\cdot)$ is a learnable linear layer and σ denotes the sigmoid function.

This design allows the model to adaptively balance episode-specific evidence and accumulated cross-episode knowledge.

3.3 Memory Update Rule

After processing episode t, the memory is updated as:

$$\mathbf{m}_{t,c} \leftarrow \mathbf{m}_{t-1,c} + \eta_t \left(\mathbf{p}_{t,c} - \mathbf{m}_{t-1,c} \right),$$

where η_t is a memory update rate that may be fixed or learned. This update can be interpreted as an exponential moving average over refined prototypes.

4. Query Classification

Given a query sample $x \in \mathcal{Q}*t$, we compute its embedding $\mathbf{z}* = f\theta(x)$ and classify it using a distance-based classifier:

$$P(y = c \mid x) = \frac{\exp(-d(\mathbf{z}, \mathbf{p}_{t,c}))}{\sum_{c'} \exp(-d(\mathbf{z}, \mathbf{p}_{t,c'}))},$$

where $d(\cdot, \cdot)$ is the squared Euclidean distance.

5. Training Objective

The episodic training objective minimizes the expected negative log-likelihood over query samples:

$$\mathcal{L}_{\text{cls}} = \mathbb{E}_{\mathcal{E}_t} \left[- \sum_{(x,y) \in \mathcal{Q}_t} \log P(y \mid x) \right].$$

To encourage memory stability, we optionally add a regularization term:

$$\mathcal{L}_{\text{mem}} = \sum_{c} \| \mathbf{m}_{t,c} - \mathbf{m}_{t-1,c} \|_2^2.$$

The final objective is:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda \, \mathcal{L}_{\text{mem}}.$$