Hierarchical Prototype Graph Networks (HPGN): Architecture Design

Ayatollah Elkolally

December 30, 2025

Abstract

This document provides a comprehensive explanation of the Hierarchical Prototype Graph Network (HPGN) architecture for few-shot hieroglyph classification, including detailed component descriptions, mathematical formulations.

1 Architecture Overview

HPGN addresses few-shot learning by combining three key innovations:

1. **Multi-scale feature extraction**: Captures both global shape and local discriminative patterns
2. **Hierarchical prototypes**: Multiple prototypes per image represent internal variation
3. **Graph-based refinement**: Cross-image reasoning via graph neural networks

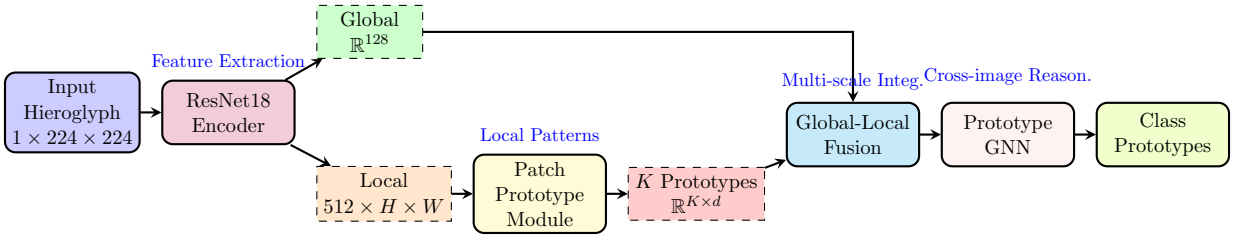

Figure 1: HPGN pipeline: Input images are processed through a ResNet encoder producing dual representations (global shape, local features), then multiple prototypes are extracted, fused with global context, and refined via graph neural networks.

2 Component Details

2.1 Component 1: ResNet Encoder

Purpose

Extract hierarchical visual features from grayscale hieroglyphs, producing both holistic shape representations and spatially-localized feature maps.

2.1.1 Architecture

We adapt ResNet18 for single-channel inputs:

$$\mathbf{x} \in \mathbb{R}^{1 \times H \times W} \quad \text{(input image)} \tag{1}$$

$$\mathbf{F}_{\text{local}} = f_{\text{ResNet}}(\mathbf{x}) \in \mathbb{R}^{512 \times H' \times W'} \quad \text{(spatial features)} \tag{2}$$

$$\mathbf{g}_{\text{global}} = \text{AdaptiveAvgPool}(\mathbf{F}_{\text{local}}) \in \mathbb{R}^{512} \tag{3}$$

$$\mathbf{g} = \text{MLP}(\mathbf{g}_{\text{global}}) \in \mathbb{R}^{d_g} \tag{4}$$

where $H' = H/32$, $W' = W/32$ due to ResNet downsampling, and $d_g = 128$ is the global embedding dimension.

2.1.2 Key Modifications

- **Input layer**: Modified `conv1` to accept 1-channel grayscale instead of 3-channel RGB
- **No pretraining**: Trained from scratch to avoid ImageNet bias
- **Dual output**: Returns both $\mathbf{g}$ (global) and $\mathbf{F}_{\text{local}}$ (local)

2.1.3 Design Rationale

- **Global embedding $\mathbf{g}$**: Captures overall hieroglyph shape (e.g., bird-like, human-like, geometric)
- **Local features $\mathbf{F}_{\text{local}}$**: Preserves spatial structure for fine-grained pattern discovery
- **ResNet18 choice**: Sufficient capacity for hieroglyphs without overfitting small datasets

Algorithm 1 ResNet Encoder Forward Pass

Require: Grayscale image $\mathbf{x} \in \mathbb{R}^{1\times224\times224}$
Ensure: Global embedding $\mathbf{g} \in \mathbb{R}^{128}$, local features $\mathbf{F} \in \mathbb{R}^{512\times7\times7}$

$\mathbf{F} \leftarrow \text{ResNet18.features}(\mathbf{x})$ ▷ Conv layers, pooling, residual blocks
$\mathbf{h} \leftarrow \text{AdaptiveAvgPool2d}(\mathbf{F})$ ▷ Global pooling: $512 \times 1 \times 1$
$\mathbf{h} \leftarrow \text{Flatten}(\mathbf{h})$ ▷ 512-dim vector
$\mathbf{g} \leftarrow \text{Linear}(\text{Dropout}(\mathbf{h}))$ ▷ Project to 128-dim
return $\mathbf{g}, \mathbf{F}$

2.2 Component 2: Patch Prototype Module

Purpose

Discover K discriminative local patterns per image using attention mechanisms, representing intra-class variation.

2.2.1 Motivation

Hieroglyphs contain multiple semantically meaningful regions (e.g., head, body, legs of an animal glyph). A single prototype cannot capture all discriminative patterns. Multiple prototypes enable:

- Modeling intra-class variation
- Part-based representation
- Robustness to occlusion or damage

2.2.2 Mathematical Formulation

Given local features $\mathbf{F} \in \mathbb{R}^{512\times H'\times W'}$:

$$\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K\} \subset \mathbb{R}^{H' \times W'} \tag{5}$$

$$\mathbf{V} = \text{Reshape}(\mathbf{F}) \in \mathbb{R}^{(H'W') \times 512} \quad \text{(flatten spatial)} \tag{6}$$

$$\mathbf{V}' = \text{MLP}(\mathbf{V}) \in \mathbb{R}^{(H'W') \times d_p} \quad \text{(project to prototype space)} \tag{7}$$

$$\mathbf{Q} \in \mathbb{R}^{K \times d_p} \quad \text{(learnable prototype queries)} \tag{8}$$

$$\mathbf{P} = \text{MultiHeadAttention}(\mathbf{Q}, \mathbf{V}', \mathbf{V}') \in \mathbb{R}^{K \times d_p} \tag{9}$$

where $d_p = 64$ is the prototype dimension and K is the number of prototypes per image.

2.2.3 Attention Mechanism

The multi-head attention computes:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{10}$$

Each prototype $\mathbf{p}_k$ is a weighted aggregation of spatial features, where weights are learned to focus on discriminative regions.

Algorithm 2 Patch Prototype Extraction

Require: Local features $\mathbf{F} \in \mathbb{R}^{512 \times H' \times W'}$, K prototypes
Ensure: Prototypes $\mathbf{P} \in \mathbb{R}^{K \times d_p}$

$\mathbf{V} \leftarrow \text{Reshape}(\mathbf{F}, (H' \cdot W', 512))$ ▷ Flatten spatial dimensions
$\mathbf{V}' \leftarrow \text{MLP}(\mathbf{V})$ ▷ Project: $(H'W', 512) \to (H'W', d_p)$
$\mathbf{Q} \leftarrow \text{LearnableParameter}(K, d_p)$ ▷ Prototype queries
$\mathbf{P}, \_ \leftarrow \text{MultiHeadAttention}(\mathbf{Q}, \mathbf{V}', \mathbf{V}')$ ▷ 2 heads
return $\mathbf{P}$

2.2.4 Hyperparameter: Number of Prototypes K

- $K = 1$: Reduces to standard prototypical network (single prototype per image)
- $K = 2$-4: Sweet spot for hieroglyphs (captures 2-4 key regions)
- $K \geq 8$: Risk of overfitting, diminishing returns

2.3 Component 3: Global-Local Fusion

Purpose

Integrate holistic shape information (global) with fine-grained local patterns (prototypes) for comprehensive representation.

2.3.1 Problem

Local prototypes alone may miss overall structure. Global embedding alone may miss fine details. Fusion combines strengths of both.

2.3.2 Fusion Strategies

We implement three fusion mechanisms:

1. Concatenation Fusion

$$\mathbf{p}_k^{\text{fused}} = [\mathbf{p}_k \,||\, \mathbf{g}] \in \mathbb{R}^{d_p+d_g} \tag{11}$$

Pros: Simple, no added parameters. *Cons*: Increases dimensionality.

2. Gated Fusion

$$\boldsymbol{\alpha}_k = \sigma\left(\mathbf{W}_g[\mathbf{p}_k \,||\, \mathbf{g}]\right) \in [0,1]^{d_p} \tag{12}$$

$$\mathbf{g}' = \mathbf{W}_t\mathbf{g} \in \mathbb{R}^{d_p} \tag{13}$$

$$\mathbf{p}_k^{\text{fused}} = \boldsymbol{\alpha}_k \odot \mathbf{p}_k + (1-\boldsymbol{\alpha}_k) \odot \mathbf{g}' \tag{14}$$

where σ is sigmoid, $\odot$ is element-wise product. The gate $\boldsymbol{\alpha}_k$ learns to adaptively blend local and global information.

Pros: Learnable, dimension-preserving. *Cons*: Adds parameters.

3. MLP Fusion

$$\mathbf{p}_k^{\text{fused}} = \text{MLP}([\mathbf{p}_k \,||\, \mathbf{g}]) \in \mathbb{R}^{d_p} \tag{15}$$

where MLP is a 2-layer feedforward network with ReLU activation.

Pros: Non-linear combination. *Cons*: Most parameters, potential overfitting.

Figure 2: Three fusion strategies for combining global shape embedding with local prototypes.

2.4 Component 4: Prototype Graph Neural Network

Purpose

Refine prototypes by leveraging cross-image similarities through graph-based message passing.

2.4.1 Motivation

Prototypes from similar hieroglyphs should reinforce each other. For example, two bird hieroglyphs may have similar beak prototypes that can be mutually refined. GNNs enable this cross-image reasoning.

2.4.2 Graph Construction

Given N support images with K prototypes each, we have NK prototype nodes:

$$\mathcal{V} = \{\mathbf{p}_k^{(i)}\}_{i=1,k=1}^{N,K} \quad \text{(nodes: all prototypes)} \tag{16}$$

$$\mathcal{E} = \{(\mathbf{p}_i, \mathbf{p}_j) \mid \mathbf{p}_j \in \text{kNN}(\mathbf{p}_i, k)\} \quad \text{(edges: k-nearest neighbors)} \tag{17}$$

We connect each prototype to its k nearest neighbors based on Euclidean distance:

$$\text{kNN}(\mathbf{p}_i, k) = \arg\min_{S \subset \mathcal{V}, |S|=k} \sum_{\mathbf{p}_j \in S} \|\mathbf{p}_i - \mathbf{p}_j\|_2 \tag{18}$$

2.4.3 Graph Attention Network (GAT)

GAT computes attention-weighted aggregation from neighbors:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{p}_i \,||\, \mathbf{W}\mathbf{p}_j]\right)\right)}{\sum_{\mathbf{p}_k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{p}_i \,||\, \mathbf{W}\mathbf{p}_k]\right)\right)} \tag{19}$$

$$\mathbf{p}'_i = \sigma\left(\sum_{\mathbf{p}_j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{p}_j\right) + \mathbf{p}_i \tag{20}$$

where $\mathbf{a}$ and $\mathbf{W}$ are learnable parameters, $\mathcal{N}(i)$ is the neighborhood of node i, and we add a residual connection.

Algorithm 3 Prototype GNN Refinement

Require: Support prototypes $\{\mathbf{P}^{(i)}\}_{i=1}^N$ where $\mathbf{P}^{(i)} \in \mathbb{R}^{K \times d_p}$
Require: Labels $\{y_i\}_{i=1}^N$
Ensure: Refined prototypes $\{\mathbf{P}'^{(i)}\}_{i=1}^N$
Flatten: $\mathcal{V} \leftarrow \{\mathbf{p}_k^{(i)}\}_{i,k}$ ▷ NK nodes
Compute pairwise distances: $\mathbf{D}_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|_2$
for each node i **do**
    Find k-nearest neighbors: $\mathcal{N}(i) \leftarrow \text{kNN}(\mathbf{p}_i, k)$
end for
Construct edge index: $\mathcal{E} \leftarrow \{(i, j) \mid j \in \mathcal{N}(i)\}$
Apply GAT: $\mathcal{V}' \leftarrow \text{GAT}(\mathcal{V}, \mathcal{E})$
Reshape back: $\{\mathbf{P}'^{(i)}\} \leftarrow \text{Reshape}(\mathcal{V}', (N, K, d_p))$
return $\{\mathbf{P}'^{(i)}\}$

2.4.4 Hyperparameter: Number of Neighbors k

- $k = 2$: Sparse graph, local refinement only
- $k = 3$-5: Balanced connectivity
- $k > 10$: Dense graph, may introduce noise

2.5 Component 5: Classification

2.5.1 Class Prototype Formation

For each class $c \in \{1, \ldots, N_{\text{way}}\}$, aggregate refined prototypes:

$$\bar{\mathbf{p}}_c = \frac{1}{K \cdot |\mathcal{S}_c|} \sum_{i \in \mathcal{S}_c} \sum_{k=1}^{K} \mathbf{p}_k'^{(i)} \tag{21}$$

where $\mathcal{S}_c$ is the set of support samples for class c.

2.5.2 Query Classification

For a query image $\mathbf{x}^q$:

$$\bar{\mathbf{p}}^q = \frac{1}{K}\sum_{k=1}^{K} \mathbf{p}_k'^q \quad \text{(average query prototypes)} \tag{22}$$

$$d_c = \|\bar{\mathbf{p}}^q - \bar{\mathbf{p}}_c\|_2 \quad \text{(Euclidean distance to class } c\text{)} \tag{23}$$

$$P(y = c \mid \mathbf{x}^q) = \frac{\exp(-d_c)}{\sum_{c'=1}^{N_{\text{way}}} \exp(-d_{c'})} \quad \text{(softmax over distances)} \tag{24}$$

2.5.3 Alternative: Ensemble Classification

When using ensemble mode, maintain separate global and local classification paths:

$$\text{logits}_{\text{global}} = -\|\mathbf{g}^q - \bar{\mathbf{g}}_c\|_2 \quad \forall c \tag{25}$$

$$\text{logits}_{\text{local}} = -\|\bar{\mathbf{p}}^q - \bar{\mathbf{p}}_c\|_2 \quad \forall c \tag{26}$$

$$\text{logits}_{\text{final}} = \alpha \cdot \text{logits}_{\text{global}} + (1 - \alpha) \cdot \text{logits}_{\text{local}} \tag{27}$$

where $\alpha \in [0, 1]$ is a learnable weight.