Multi-Modal Knowledge-Primed Neural Networks (MM-KPNNs)

Purpose & Biological Context

This framework guides you through developing a Graph Neural Network (MM-KPNN) pipeline that integrates single-cell multi-omic data—combining scRNA-seq and scATAC-seq—into a unified graph-based framework for immune cell-type classification. By fusing gene expression and chromatin accessibility into node features and leveraging k-NN graphs to model cellular similarity, MM-KPNN captures molecular, regulatory, and phenotypic relationships, enabling robust and interpretable predictions.

How to Read and Reproduce the MM-KPNN Pipeline

The pipeline includes concise rationale and interpretation comments explaining each step's purpose and results. Code is highly commented and modular, with figures embedded directly in the workflow, allowing users to visualize outputs step by step and ensure clarity, traceability, and reproducibility across environments.

Pipeline Overview

The workflow proceeds step by step—from data preprocessing to model training, interpretability, and biological extension.

Step 1. Imports & Environment Setup

- Load core libraries: **PyTorch**, **PyTorch-Geometric**, **Scanpy**, **scikit-learn**, **UMAP**, **Matplotlib**.
- Define random seeds and working directories.

Environment Setup

Library	Version	Purpose
Python	3.9.16	Language runtime
PyTorch (+cpu)	2.7.0+cpu	Core deep learning framework

PyTorch-Geometric	2.6.1	Graph Neural Network layers & utilities
Scanpy	1.9.1	Single-cell data I/O & preprocessing
scikit-learn	1.0.2	k-NN, train/test splits, and metrics
umap-learn	0.5.3	Embedding visualization (UMAP projections)
Matplotlib	3.5.3	Plotting and figure generation

Step 2. Model Definition

- Define **MM-KPNN** with two **GCNConv** layers and a **linear head** for cell-type classification.
- The layered graph structure propagates features across local neighborhoods to integrate both regulatory and phenotypic context.

Step 3. Dataset Loading & Preprocessing

- Load example data: `scanpy.datasets.pbmc3k_processed()`.
- Select highly variable genes (HVGs) and compute PCA coordinates.
- Build a **k-NN graph** → `edge_index`.
- Split data (80/10/10) into **train**, **validation**, and **test** indices.

Step 4. Model Instantiation & Device Setup

- Detect compute device (**cuda/cpu**) and move model + tensors to the selected device.

Step 5. Loss & Metrics

- **Loss:** CrossEntropyLoss().
- **Metrics:** Accuracy, precision, recall, and macro-F1 (via scikit-learn).

Step 6. Training & Validation Loop

- Standard forward/backward pass:
  `optimizer.zero_grad()` → forward → `loss.backward()` → `optimizer.step()`.
- Log training and validation metrics across epochs.

Step 7. Test Set Evaluation

- Evaluate model on held-out data.
- Report **test loss**, **accuracy**, and **macro-F1** for each cell type.

Step 8. Input-Level Interpretability

- Apply **gradient-based feature saliency** to identify genes or peaks most influential for predictions.
- Visualize class-wise feature distributions to confirm that top-ranked features correspond to biologically relevant markers.

Step 9. Performance Evaluation & Embedding Visualization

- **UMAP** projection of node embeddings → visualize cluster separation.
- **Silhouette Score** → quantify intra-class cohesion vs. inter-class separation.
- **Confusion Matrix** → display per-class accuracy and misclassification patterns.

Step 10. Swap in Real Data (Optional)

- Replace demo data with your own **scRNA + scATAC** dataset.
- Preprocess (normalization, PCA/LSI, TF-IDF), build a new **k-NN graph**, and stratify into splits.
- All downstream steps—training, evaluation, interpretability—remain identical.

Step 11. Interpretable Representation Analysis & Comparative Evaluation

- **Bottleneck-Level Attribution (conv2 activations):** Identify pathway/TF-like activity patterns per class.
- **Bottleneck → Class Connection Importance (fc weights):** Determine which latent programs drive class predictions.
- **Attribution Stability Check:** Compare saliency across random seeds.
- **Baseline Embedding Comparison (PCA):** Classical reference showing lower separability.

Step 12. Technical Recommendations & Environment Summary

- Provide guidance on **reproducibility**, **parameter tuning**, and **graph/model design**.

Step 13. Biological Expansion

- Extend MM-KPNN to integrate **transcription factor–target priors**, iterative **troubleshooting**, and **spatial transcriptomics** for modeling cell–cell communication and tissue context.