CS224W Project Milestone Report

Graph Neural Networks for Stock Market Prediction and Portfolio Management

Author: Tianhui Huang, Haonan Zhu

1 Milestone Significance & Strategy

1.1 Deliverable Focus

1. Demonstrate the core code and data artifacts for Phases 1–4, showing that the pipeline runs **end-to-end** with none credit running.
2. Report the current training status and metrics (with the caveat that this run uses synthetic data).
3. Provide a draft report covering problem motivation, data processing, model design, and debugging notes.

1.2 High-Level Technical Roadmap

- **Phases 1–2 (Data & Graph):** Build a multi-modal financial knowledge graph by generating node features (technical, fundamental, sentiment, macro) and heterogeneous edges (rolling correlation, fundamental similarity, static relations).
- **Phase 3 (Baseline):** Use a **"flattened" GNN baseline (GAT + GCN mix)** to validate graph inputs and labels sanity.
- **Phase 4 (Core Model):** Adopt a **Role-Aware Graph Transformer** (RelationAware GATv2 + PEARL positional embedding) to capture relation-specific semantics and meet explainability needs.

1.3 Modeling Rationale

The two-stage strategy (simple GNN $\rightarrow$ Transformer) is chosen to isolate data/feature issues from model capacity issues. The core model focuses on heterogeneous relations because: pure time-series models ignore cross-equity interactions, table models require heavy manual feature engineering, and simple homogeneous GNNs suffer from edge-weight dilution. The overall Goal is to validate heterogeneous relations before preparing for Phase 5 **reinforcement learning (RL)**.

2 Problem Description & Motivation

Financial markets exhibit rich cross-equity relationships that traditional time-series models fail to capture. GNNs enable message passing in heterogeneous networks, aiding in reasoning about systemic risk and interdependencies.

1. Use GNNs to exploit relational signals across equities.
2. Improve recall of **"down" events** for risk management.
3. Apply CS224W graph techniques to a temporal financial graph, evaluating strengths and limitations.

3 Dataset Description & Processing (Phase 1 & Phase 2)

Note: Synthetic OHLCV data for 50 stocks (2018–2024) is used to validate code and training stability.

3.1 Data Collection & Feature Engineering

3.1.1 Price (OHLCV) and Technical Indicators

- **Price:** Synthetic data based on **Geometric Brownian Motion** (GBM) for validation purposes.
- **Technical Indicators (10 × 50 stocks):** Log returns ($\text{LogRet}_{1,5,20}$), 30-day volatility (Vol_{30d}), RSI-14, MACD components, Bollinger Band Width (BB_Width), Price-to-SMA50, ATR-14, OBV, MFI. These form the primary **node features** (707 columns).

3.1.2 Fundamental / Sentiment / Macro Features

- **Features:** PE, PB, ROE, Debt/Equity, VIX, risk_free_rate.
- **Processing:** Offline mode uses AR(1) or Ornstein–Uhlenbeck processes; real-mode uses downloads/perturbations. Data is cleaned ($\log(1+x)$), filled, and scaled via `StandardScaler`.

3.2 Dynamic and Static Edge Computation

- **Static Relations:** Industry, Supply Chain (directed), Competitors (undirected), ensuring connectivity.
- **Dynamic Edges:**
  1. **Rolling Correlation:** 30-day log return correlation, retained if $|\rho| \geq 0.6$, subject to **Top 5 neighbors pruning**.
  2. **Fundamental Similarity:** Cosine similarity on Z-scored fundamentals, retained if ≥ 0.8.

3.3 Heterogeneous Graph Construction (Phase 2)

- **Pipeline:** Align all features/edges → map tickers to indices → generate dynamic/static edges → apply Top-K sparsification → build daily `HeteroData` objects (`graph_t_YYYYMMDD.pt`).
- **Quality Checks:** Confirmed average degree 13.8 (max 21, min 6); 1,778 graphs match trading days; normalized weights $\in [0, 1]$.

4 Model Design & Evaluation Workflow (Phase 3 & Phase 4)

4.1 Phase 3 – Baseline GNN (Flattened Graph)

- **Input Preparation:** Each day produces one homogeneous `Data` graph; node features are 707-dim; edge weights are min-max normalized.
- **Architecture:** `GATConv(707 → 128, heads=4)` followed by `GCNConv(512 → 128)`, classified by a light two-layer MLP head.
- **Training: Focal Loss** ($\alpha = 0.5, \gamma = 2.0$), Adam (LR 5e−4), `ReduceLROnPlateau`, Early Stopping (patience 5).
- **Rationale:** Confirms features/labels behave sensibly; provides benchmark for future comparisons; fast to train ($\sim$ 3.8s/epoch on CPU).

4.2 Phase 4 – Role-Aware Graph Transformer (Heterogeneous Graph)

- **Motivation:** Needs relation-specific parameters to avoid semantic dilution; attention scores provide explainability; **PEARL positional embeddings** inject structural context.
- **Architecture:**
  1. Input: daily `HeteroData` with PEARL embedding (32-dim) concatenated to 707-dim features.
  2. Transformer Stack (2 layers): `HeteroConv` with 5× `RelationAwareGATv2Conv` per edge type.
  3. Aggregation: Learned attention vector ($\mathbf{w} \in \mathbb{R}^{128\times5}$) softmaxed across relation outputs.
- **Training & Evaluation:** Identical loss/optimizer/scheduler to Phase 3; logs per-relation attention weights; current run: F1 ≈ 0.65 (near random, as expected).
- **Planned Analyses:** Compare relation-aware vs flattened baselines; analyze attention weights during market events; extend with temporal encoders (GRU/LSTM).

4.3 Evaluation Pipeline Summary

The workflow loads/generates daily graphs, trains Phase 3 and Phase 4 models on identical chronological splits, exports metrics, confusion matrices, ROC curves, and relation attention statistics, and compares runs on a unified dashboard. Profitable and calibration metrics will be added once real data is available.

5 Code Deliverables & Program Structure

5.1 Dataset Processing (Processing the Dataset)

- `scripts/phase1_data_collection.py`: Fetches raw OHLCV, macro, sentiment data; synthesizes fallback series when offline.
- **Key Outputs:** `data/raw/stock_prices_ohlcv_raw.csv`, `data/raw/sentiment_macro_raw.csv`. Supports resume and logging.
- `scripts/phase1_static_data_collection.py`: Generates static relations (industry, supply chain, competitors).
- **Key Outputs:** `data/raw/static_sector_industry.csv`, `data/raw/static_supply_competitor_edges.csv`. SPY Top-50 fallback ensures connectivity.
- `scripts/phase1_feature_engineering.py`: Computes technical indicators, normalizes fundamentals/sentiment, produces node feature matrix and edge parameters.
- **Key Outputs:** `data/processed/node_features_X_t_final.csv`, `data/edges/edges_dynamic_corr_params.pkl`, etc.
- `scripts/phase2_graph_construction.py`: Builds daily heterogeneous graphs with Top-K sparsification and normalization.
- **Key Output:** `data/graphs/graph_t_YYYYMMDD.pt`. Includes performance logs (~ 0.12s/day).

Usage Example: `python scripts/phase1_data_collection.py -tickers config/tickers_spy50.txt`

5.2 Model Training / Evaluation (Training/Evaluating the Model)

- `scripts/phase3_baseline_training.py`: Trains/evaluates the flattened GNN baseline.
- **Key Outputs:** `models/checkpoints/`, TensorBoard runs, `metrics_phase3.csv`. Supports CLI options for epochs, loss-type, device.
- `scripts/phase4_core_training.py`: Trains/evaluates the Role-Aware Graph Transformer.
- **Key Outputs:** `models/core_transformer_model.pt`, attention weights logs, `metrics_phase4.csv`. Supports CLI options for hidden-dim, layers, AMP, grad-clip.
- `milestone/METRICS_QUICK_REFERENCE.md`: Reference for metric formulas, interpretations, and debugging tips for review sessions.

Typical Workflow: `python scripts/phase3_baseline_training.py -epochs 40 -loss-type focal`

5.3 Additional Utilities (Any Other Programs Required)

- `scripts/phase4_hyperparameter_sweep.py`: Orchestrates grid search; stores per-run configs, metrics, and checkpoints under `models/sweeps/`.
- `scripts/utils_data.py`: Shared helpers for loading graphs, batching, and sanity checks.
- `runs/` & `models/`: Standardized directories for TensorBoard summaries, checkpoints, and logs.

Runtime & Artifacts: CPU runs take $\sim$ 4s/epoch (Phase 3) and $\sim$ 28s/epoch (Phase 4). GPU usage is recommended.

6 Current Metrics & Assessment

6.1 Synthetic Data Summary

Synthetic data results **validate the pipeline** (metrics near random):

- Phase 3 (Baseline): Val F1 ≈ 0.67, ROC-AUC ≈ 0.51.
- Phase 4 (Transformer): Test F1 ≈ 0.65, ROC-AUC ≈ 0.50.

These numbers confirm the code runs and training stabilizes; they do **not** imply predictive power, as synthetic prices lack structural signal.

6.2 Why These Metrics?

- **F1 Score (macro/micro)**: Balances precision/recall, critical for the high-cost "down" class.
- **ROC-AUC**: Threshold-independent ranking ability.
- **Confusion Matrix**: Reveals FP/FN patterns to drive subsequent strategy decisions (e.g., increasing down-class recall).
- **Planned Future Metrics**: Cumulative return, max drawdown, Sharpe Ratio, Downside Deviation, Brier Score, and calibration curves once real data is available.

6.3 Real-Data Plan & Next Steps

1. Rerun Phase 1 with **real data**, align calendars, and handle suspensions/splits/missing values.
2. Extend metrics with profitability and calibration measures.
3. Examine precision/recall trade-offs for the down class.
4. Tune Top-K or adopt MST-based sparsification if graphs become too dense or sparse.

6.4 Risks & Mitigation

1. **Data Quality:** Real anomalies (e.g., stock splits) may distort correlations; plan careful cleaning and backfilling.
2. **Graph Sparsity vs. Over-Smoothing:** Maintain Top-K constraint around 5; dynamically adjust with real density statistics.
3. **Model Architecture:** Evaluate alternative convolutions (SAGEConv); analyze transformer attention weights for interpretability; consider temporal encoders (GRU/LSTM).
4. **Training Practices:** Consider Warmup + ReduceLROnPlateau; adopt `NeighborSampler` on GPU or with credit.