Multi-Graph Meta-Transformer: An Interpretable Framework for Cross-Graph Functional Alignment in Neural Decoding

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neuroscience experiments often capture brain signals from heterogeneous individuals, each with unique neural dynamics, even in response to the exact same stimuli. This subject-to-subject variability makes it challenging to aggregate data and extract common neural patterns. To address this, we propose Multi-Graph Meta-Transformer (MGMT), a unified framework that learns from a set of graphs sharing a single prediction target, while respecting their individual structures. MGMT captures graph-specific patterns, aligns their structural representations in a shared latent space, and integrates them to learn a robust and generalizable structure. Conceptually, MGMT reframes graph fusion as functional alignment, borrowing statistical power by linking regions that exhibit similar patterns across graphs. We apply MGMT to analyze hippocampal local field potentials (LFPs) from five rats performing an odor–sequence task, where the neural activity of each rat is represented by a distinct graph. MGMT uses Graph Transformer encoders to identify *supernodes* and then builds a *meta-graph* by forming *superedges* across graphs based on similarities of latent node representation. This restricts message passing to only functionally aligned pairs, reducing cross-graph noise and yielding more accurate, interpretable graph-level predictions. In our neural decoding experiment, MGMT outperforms existing fusion strategies. Notably, it uncovers distal CA1 selectivity for non-spatial information and demonstrates that its learned inter-graph connections capture meaningful brain dynamics.

1 Introduction

Graphs are fundamental data structures in many domains including neuroscience [1], social networks [2, 3] and molecular biology [4, 5, 6]. While powerful models like Graph Neural Networks (GNNs) [7, 8, 9] and the more recent Graph Transformers (GTs) [10, 11, 12, 13] excel at learning from single graphs, many real-world problems require integrating information across multiple heterogeneous graphs. For instance, neuroscience experiments studying brain dynamics often generate graphs from multiple subjects, each with distinct connectivities and node sets [1]. Enhancing prediction performance or extracting common neural patterns in such settings requires a framework that can effectively integrate these disparate graphs. However, how to best adapt powerful architectures like the GT for this multi-graph integration challenge remains underexplored. Existing fusion paradigms fall short as they either assume a single, unified graph with aligned nodes [14, 15], or they collapse each graph's topology into a single vectorized embedding before fusion [16, 17]. Consequently, valuable structural information both within and between the graphs is lost.

To address this gap, we propose **Multi-Graph Meta-Transformer (MGMT)**, a novel framework designed to fuse information from collections of heterogeneous graphs. Our unified framework,

Submitted to the AI for Science workshop (NeurIPS 2025). Do not distribute.

which we group under the umbrella term "multi-graph," is broadly applicable and handles several common scenarios including: **multi-modal** (graphs from different measurement channels, e.g., MRI vs. clinical UDS), **multi-view** (different structural views of the same data, e.g., different feature subsets or data measured under different conditions), and **multi-subject** (graphs from different subjects in the same experiment). Our approach involves independently processing each graph (modality/view/subject) using dedicated GT encoders, resulting in intra-graph representations that are mapped into a shared latent space. It then integrates these representations by constructing a *meta-graph*. This is achieved by identifying the most informative *supernodes* within each graph through attention mechanism and connecting them with *superedges* based on similarity in their learned latent embeddings. By applying additional GT layers to this meta-graph, MGMT facilitates selective information sharing between functionally aligned nodes across the collection, enabling the joint learning of both local and global patterns.

Key Contributions

We introduce MGMT, a novel framework for multi-graph fusion. It learns robust graph-specific representations through dynamic aggregation of GT layers at varying depths. It further constructs a meta-graph to enable selective, structured information sharing across graphs in the latent space.

The framework provides inherent interpretability through its meta-graph construction. The identified supernodes highlight influential, task-relevant subgraph structures, while the learned superedges pinpoint functional alignments between graphs, offering a clear explanation of cross-graph interactions.

We also provide a comprehensive theoretical study that analyzes both intra-graph and inter-graph properties of MGMT, offering rigorous analysis on its representational capabilities.

Finally, we demonstrate MGMT's effectiveness on challenging neuroscience datasets, where it successfully extracts meaningful neuronal activity patterns shared across subjects. These findings are validated by existing interdisciplinary research, showcasing the model's potential for real-world scientific discovery.

Related Work

Graph Representation Learning Graph Neural Network (GNN) is the cornerstone of modern graph machine learning. It learns node representations by iteratively aggregating features from local neighbors through message-passing [7, 8, 18]. To better capture long-range dependencies and enhance expressive power, Graph Transformers (GTs) have emerged as a powerful alternative. These models adapt the global self-attention mechanism, originally from natural language processing [19], for graph-structured data, typically by injecting structural information through positional encodings or by combining attention with message-passing components [20, 11, 21, 12]. While both architectures are highly effective for single-graph tasks, they are not inherently designed to fuse information from a collection of multiple, potentially heterogeneous graphs.

Multimodal and Heterogeneous Graph Learning A distinct line of research that may appear similar is multimodal or heterogeneous graph learning. However, its problem setting is fundamentally different from our multi-graph fusion task. These methods operate on a single, unified graph that integrates various data types. For example, frameworks like UniGraph2 [14] and HetGNN [15] assume a single graph where node possess multiple features types from different modalities, such as text or images. This assumption collapses multiple data sources into one large graph. Other works, such as MMGL [22], construct a single population-level graph where nodes represent subjects, and features from all modalities are concatenated before graph construction. While effective for their intended purpose, these methods are not applicable to the more general and challenging problem of fusing a collection of graphs with distinct, unaligned node sets, which is the focus of our work.

General-Purpose Multimodal Fusion General-purpose frameworks including MultiMoDN [23], FlexCare [17], MedFuse [16], and Meta-Transformer (MT) [24] considers integration of multiple modalities, including graphs or images. One could technically apply these frameworks to a multi-graph fusion problem by treating each graph as a separate modality. These frameworks, such as MedFuse [16] typically use modality-specific encoders to first transform each input into a single latent vector. For a graph, this means collapsing its entire topological structure into one embedding. These vectors are then fused with operations such as concatenation for the down stream tasks. This process

not only discards the rich structural information within each graph but also offers no mechanism for modeling the fine-grained, structural relationships between graphs, highlighting the need for a truly graph-native fusion methodology.

2 Methodology

In this section, we present MGMT, detailing its prediction pipeline based on GTs and meta-graph construction, followed by describing how to interpret MGMT by identifying significant nodes and edges in Section 2.2. An overview of the entire framework is provided in Figure 1.

2.1 Multi-Graph Meta-Transformer (MGMT)

MGMT fuses multi-graph data using hierarchical meta-graph modeling through the following steps:

2.1.1 Graph-Specific Transformer Encoders

For each instance, we observe a collection of n graphs. For $i = 1, \ldots, n$, we denote the graph as $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ with node set $\mathcal{V}_i$ of size $N_i = |\mathcal{V}_i|$, and edge set $\mathcal{E}_i$. Each graph $\mathcal{G}_i$ is characterized by a node feature matrix $\boldsymbol{X}_i \in \mathbb{R}^{N_i \times d}$ and an adjacency matrix $\boldsymbol{A}_i \in \{0,1\}^{N_i \times N_i}$. Graphs per each instance may differ in size and structure (for presentation purpose only, we assume feature size is d across all graphs), yet the collection $\{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ share a common label $Y \in \mathcal{Y}$. The task is graph-level classification of the shared label Y using evidence aggregated across graphs. Throughout this paper, we use bold uppercase letters (e.g., $\boldsymbol{X}$) for matrices and and bold lowercase letter (e.g., $\boldsymbol{x}$) for vectors, and $[n]$ denoting the set $\{1, \ldots, n\}$.

We formalize the core graph-specific Transformer mechanics used in MGMT, building upon the localized graph-aware attention principles detailed in Appendix A1. For each $i \in [n]$, the graph $\mathcal{G}_i$ with node features $\boldsymbol{X}_i \in \mathbb{R}^{N_i \times d}$ undergoes L GT layers with multi-head self-attention. Starting with $\boldsymbol{H}_i^{(0)} = \boldsymbol{X}_i$ as initial features, we define the extended neighborhood $\bar{\mathcal{N}}(u) = \mathcal{N}(u) \cup \{u\}$ to ensure nodes attend to themselves during message passing.

For layer $\ell \in [L]$, attention head $m \in [M]$, and edge $(u, v) \in \mathcal{E}_i \cup \{(u, u)\}$, we compute:

$$\begin{aligned} \boldsymbol{Q}_{i,u}^{(\ell,m)} &= \boldsymbol{W}_{Q,i}^{(\ell,m)} \boldsymbol{H}_{i,u}^{(\ell-1)} + \boldsymbol{b}_{Q,i}^{(\ell,m)}, \\ \boldsymbol{K}_{i,v}^{(\ell,m)} &= \boldsymbol{W}_{K,i}^{(\ell,m)} \boldsymbol{H}_{i,v}^{(\ell-1)} + \boldsymbol{b}_{K,i}^{(\ell,m)}, \\ \boldsymbol{V}_{i,v}^{(\ell,m)} &= \boldsymbol{W}_{V,i}^{(\ell,m)} \boldsymbol{H}_{i,v}^{(\ell-1)} + \boldsymbol{b}_{V,i}^{(\ell,m)}, \end{aligned} \qquad \begin{aligned} \alpha_{i,uv}^{(\ell,m)} &= \frac{\exp\left(\boldsymbol{Q}_{i,u}^{(\ell,m)\top} \boldsymbol{K}_{i,v}^{(\ell,m)} / \sqrt{d'}\right)}{\sum_{v' \in \bar{\mathcal{N}}(u)} \exp\left(\boldsymbol{Q}_{i,u}^{(\ell,m)\top} \boldsymbol{K}_{i,v'}^{(\ell,m)} / \sqrt{d'}\right)}, \\ \boldsymbol{Z}_{i,u}^{(\ell,m)} &= \sum\nolimits_{v \in \bar{\mathcal{N}}(u)} \alpha_{i,uv}^{(\ell,m)} \boldsymbol{V}_{i,v}^{(\ell,m)}, \end{aligned} \tag{1}$$

where $\boldsymbol{H}_{i,u}^{(\ell-1)} \in \mathbb{R}^d$ is the feature of node u at layer $\ell - 1$, $d' = d/M$ denotes the per-head dimension. Projection matrices $\boldsymbol{W}_{Q,i}^{(\ell,m)}, \boldsymbol{W}_{K,i}^{(\ell,m)}, \boldsymbol{W}_{V,i}^{(\ell,m)} \in \mathbb{R}^{d' \times d}$ and biases $\boldsymbol{b}_{Q,i}^{(\ell,m)}, \boldsymbol{b}_{K,i}^{(\ell,m)}, \boldsymbol{b}_{V,i}^{(\ell,m)} \in \mathbb{R}^{d'}$ are learnable parameters. The query vector $\boldsymbol{Q}_{i,u}^{(\ell,m)}$ represents information node u seeks from neighbors, key vector $\boldsymbol{K}_{i,v}^{(\ell,m)}$ encodes neighbor v's relevance, and value vector $\boldsymbol{V}_{i,v}^{(\ell,m)}$ contains content to be aggregated. Attention score $\alpha_{i,uv}^{(\ell,m)}$ determines how much node u attends to node v.

The outputs of all heads are concatenated ($\|$ denotes the concatenation) and transformed via:

$$\boldsymbol{Z}_{i,u}^{(\ell)} = \Big\|_{m \in [M]} \left[\boldsymbol{Z}_{i,u}^{(\ell,1)}, \ldots, \boldsymbol{Z}_{i,u}^{(\ell,M)}\right] \boldsymbol{W}_{O,i}^{(\ell)} + \boldsymbol{b}_{O,i}^{(\ell)},$$

where $\boldsymbol{W}_{O,i}^{(\ell)} \in \mathbb{R}^{d \times d}$, $\boldsymbol{b}_{O,i}^{(\ell)} \in \mathbb{R}^d$. Stacking these vectors across all nodes yields $\boldsymbol{Z}_i^{(\ell)} \in \mathbb{R}^{N_i \times d}$.

This attention-based aggregation $\boldsymbol{Z}_i^{(\ell)}$ then passes through feedforward network with activation, residual connection, and layer normalization to produce the final output $\boldsymbol{H}_i^{(\ell)}$. Specifically:

$$\boldsymbol{H}_i^{(\ell)} = \text{LayerNorm}(\boldsymbol{Z}_i^{(\ell)} + \sigma(\text{FFN}(\boldsymbol{Z}_i^{(\ell)}))) \tag{2}$$

After L layers, we obtain final output and attentions by dynamically aggregating across all depths:

$$\boldsymbol{H}_i = \sum\nolimits_{\ell\in[L]} \Gamma^{(\ell)}\, \boldsymbol{H}_i^{(\ell)} \in \mathbb{R}^{N_i\times d},$$
$$\boldsymbol{\alpha}_i = \left\{ \alpha_{i,uv} = \sum\nolimits_{\ell\in[L]} \Gamma^{(\ell)} \left(\frac{1}{M} \sum\nolimits_{m\in[M]} \alpha_{i,uv}^{(l,m)} \right) \right\}_{(u,v)\in\mathcal{E}_i\cup\{(u,u)\}}, \tag{3}$$

where $\{\Gamma^{(\ell)}\}_{l=1}^{n}$ are confidence scores measuring the quality of each Transformer layer (see Appendix A2 for computation details).

2.1.2 Super-node Extraction

To identify the most informative nodes in each graph i, we extract *super-nodes* based on the learned attention scores $\boldsymbol{\alpha}_i$ in (3). Given a predefined threshold τ, we form the set of super-nodes as

$$\mathcal{S}_i = \left\{ u \in \mathcal{V}_i \,\middle|\, \sum\nolimits_{(u,v)\in\mathcal{E}_i} \alpha_{i,uv} \geq \tau \right\}. \tag{4}$$

Intuitively, $\sum_{(u,v)\in\mathcal{E}_i} \alpha_{i,uv}$ quantifies the total attention distributed by node u to its neighbors.

We then induce a subgraph over these nodes:

$$\mathcal{G}_i' = (\mathcal{S}_i, \mathcal{E}_i'), \mathcal{E}_i' = \{(u,v) \in \mathcal{E}_i \mid u, v \in \mathcal{S}_i\} \tag{5}$$

Additionally, we conduct a sensitivity study in Appendix A10 to examine how choices of threshold τ influence model performance. Our analysis reveals that τ controls a trade-off: a higher τ creates a sparser meta-graph, which risks information loss, while a lower τ retains more nodes, risking overfitting to noise. In practice, by guiding the selection of τ via cross-validation, we identified a robust range of values that yields stable performance.

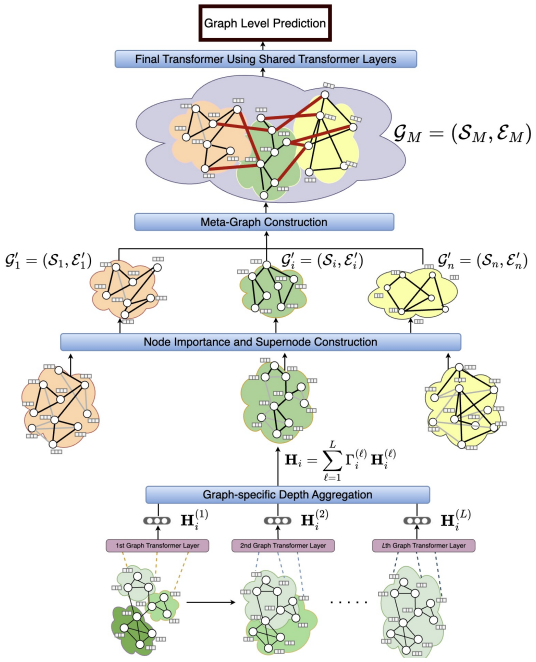

Figure 1: Architecture of the Multi-Graph Meta-Transformer (MGMT). Depth-Aware GT layers process individual graphs, extracting super-nodes to form a meta-graph. Additional GT layers model both intra- and inter-graph interactions.

2.1.3 Meta-Graph Construction

To model both intra- and cross-graph interactions, we construct an instance-level meta-graph $\mathcal{G}_M = (\mathcal{S}_M, \mathcal{E}_M)$, where $\mathcal{S}_M = \bigcup_{i=1}^{n} \mathcal{S}_i$ contains all graph-specific super-nodes. Each node $u \in \mathcal{S}_i$ is associated with a latent embedding $\boldsymbol{H}_{i,u} \in \mathbb{R}^d$ as defined in (3).

The edge set $\mathcal{E}_M$ of the meta-graph includes two components. First, we retain all intra-graph edges from the pruned graphs $\mathcal{G}_i' = (\mathcal{S}_i, \mathcal{E}_i')$, preserving graph-specific relationships. Second, we introduce inter-graph edges between cross-graph super-nodes based on their feature similarity. For any node pair (u, v) with $u \in \mathcal{S}_i$, $v \in \mathcal{S}_j$, and $i \neq j$, we compute the cosine similarity:

$$e_{uv} = \frac{\boldsymbol{H}_u^\top \boldsymbol{H}_v}{\|\boldsymbol{H}_u\| \|\boldsymbol{H}_v\|} \tag{6}$$

If the similarity score e_{uv} exceeds a predefined threshold γ, the edge (u, v) is added to $\mathcal{E}_M$.

The resulting adjacency matrix $\boldsymbol{A}_M \in \mathbb{R}^{|\mathcal{S}_M|\times|\mathcal{S}_M|}$, encodes both intra- and inter-graph relationships among super-nodes. A widely adopted assumption for graph signals is that values change smoothly across adjacent nodes [25]. MGMT applies this at the meta-graph level: superedges connect only supernodes with similar embeddings, promoting aligned message passing.

As shown in Appendix A10, accuracy is typically non-monotone in γ, reflecting the trade-off between dense connectivity (risking overfitting/noisy exchanges) and sparsity (losing cross-graph interactions), In practice, γ is selected on a validation split.

Finally, in Appendix A11, we compared cosine similarity with Pearson correlation, Euclidean distance, and dot product for defining inter-graph edges. The results show that performance remains broadly robust across metrics, suggesting that our framework is not sensitive to the choice of similarity metric.

2.1.4 Feature Learning and Prediction

After constructing meta-graph $\mathcal{G}_M$, we apply additional GT layers to the stacked super-node embeddings $\boldsymbol{H}_M^{(0)} \in \mathbb{R}^{|\mathcal{S}_M| \times d}$. Multi-head self-attention and feedforward updates are applied to capture global contextual dependencies, resulting in updated super-node embeddings $\boldsymbol{H}_M \in \mathbb{R}^{|\mathcal{S}_M| \times d}$.

For classification, we apply permutation-invariant pooling followed by a fully connected network:

$$\hat{y} = f(\text{Pool}(\boldsymbol{H}_M)), \tag{7}$$

where $\text{Pool}(\cdot)$ can be a mean, concatenation, or attention-based function, and $f(\cdot)$ maps the pooled representation to class probabilities $\hat{y} \in \mathbb{R}^{|\mathcal{Y}|}$.

This final step enables MGMT to make robust predictions by integrating both modality-specific structures and cross-modal interactions in a unified graph representation.

2.2 Interpretation of MGMT

The identified meta-graph $\mathcal{G}_M$ is analyzed via (1) **Node-level analysis**, highlighting influential nodes and their contributions, and (2) **Edge-level analysis**, uncovering critical relationships among these nodes. This framework enhances transparency, provides actionable insights for domain experts, and is further evaluated in our neuroscience application results.

3 Theoretical Properties

In Appendix A3, we establish the theoretical foundations of MGMT through two analyses. First, our *intra-graph analysis* demonstrates the superior representational power of our approach within each graph. Specifically, we prove that MGMT's depth-aware Graph Transformers (see (1)–(3)) can capture complex L-hop feature mixing, which measures expressive capability, while standard Graph Transformers cannot. Second, our *inter-graph analysis* shows that the explicit meta-graph construction leads to enhanced predictive power compared to standard late-fusion alternatives. Complete proofs are provided in Appendix A4, with additional theoretical results in Appendix A5.

4 Numerical Experiments

We evaluate the effectiveness of MGMT on four datasets in the main paper (three synthetic + LFP) and an additional Alzheimer's case study in Appendix A8. Among these, the three synthetic datasets and Alzheimer's dataset are multi-modal (multiple modalities per sample), while the LFP dataset is multi-subject (graphs from different animals treated as distinct modalities). Our analysis is structured into two parts: (1) comparisons with a broad set of baseline models, and (2) ablation studies to assess the contribution of each component within MGMT. Performance results are summarized in Figures 2, A5, and A4, with detailed accuracy values and standard errors reported in Tables A2 and A3.

4.1 Baseline Comparisons

We compare MGMT against the following three categories of baselines:

(i) **Single-Source Models:** These models are trained on each data source independently and include Deep Neural Networks (DNNs) [26], Graph Neural Networks (GNNs), Differentiable Pooling (DiffPool) [27], standard Transformers, and Graph Transformers.

(ii) **Early Fusion Models (Feature Concatenation):** For each data source, features are extracted using source-specific architectures (e.g., DNN, GNN, DiffPool). These features are concatenated and input to a shared classifier, typically a DNN [28, 29, 30].

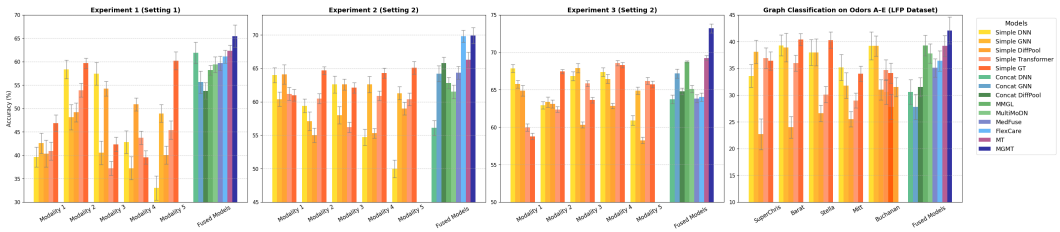

Figure 2: Average test accuracy and standard error bars across synthetic and LFP datasets. **(a)** Experiment 1 (Setting 1) uses a sample size of 100, with 5 nodes that are all informative. Experiments 2 and 3 (Setting 2) both involve structured noise; Experiment 2 uses 100 samples, and Experiment 3 uses 2,000 samples. All three experiments involve 50 nodes, of which 40 are informative. **(b)** Odor–sequence LFP decoding across five animals plus fused models. Each bar represents the average test accuracy across 5 folds, along with the corresponding standard error. Across all configurations, the proposed MGMT model achieves the best performance.

(iii) **Benchmark Fusion Models:** We evaluate MGMT against recent fusion frameworks; MMGL [22], MultiMoDN [23], FlexCare [17], MedFuse [16], and Meta-Transformer (MT) [24], each designed to integrate information from multiple input sources or feature streams.

4.2 Ablation Study

To quantify each component's impact, we evaluate five ablations: (1) removing adaptive depth selection (use final Transformer layer), (2) removing supernode selection (include all nodes in the meta-graph), (3) removing inter-modality edges, (4) removing intra-modality edges, and (5) disabling both the meta-graph and adaptive depth mechanisms, using simple late fusion of fixed-depth Transformer outputs. results can be found in Table A3 and Figure A5.

Appendix A6 provides detailed descriptions of all baseline models, multimodal fusion benchmarks, and MGMT ablation variants, along with a structured categorization of these models based on their fusion strategy, use of graph-structured modeling, attention mechanisms, and architectural novelty.

4.3 Experimental Setup

In MGMT framework, for all the datasets we use TransformerConv layers with global max or mean pooling to generate graph-level embeddings. Our models are trained on 80% of the data, with 10% reserved for validation and 10% for testing, using the Adam optimizer and early stopping based on validation loss. For real datasets, all models are trained using 5-fold cross-validation. Hyperparametersincluding the number of layers, dropout rate, learning rate, training epochs, and node importance thresholds, are optimized using Optuna with 100 trials. The best hyperparameters are selected based on validation performance. For simulation studies, models are trained and evaluated over 50 independent runs. We report the mean test accuracy and standard error across these runs.

Runtime and Scalability. Appendix A9 presents a comprehensive analysis of MGMT's efficiency using three complementary metrics: (i) theoretical time complexity of each architectural component, (ii) empirical runtime profiling across datasets including average per-epoch runtimes and stage-wise breakdowns of MGMT(e.g., encoding, supernode construction, meta-graph reasoning), and (iii) controlled scalability experiments varying graph size, modality count, sample size, and feature dimensionality. Together, these analyses confirm that MGMT achieves practical runtime efficiency and scales predictably in line with standard Transformer-based graph architectures.

4.4 Synthetic Experiments

In this section, we present a comprehensive evaluation of MGMT using synthetic datasets. We simulate graphs under varying conditions, altering the feature generation mechanisms, the number of nodes N, the sample size n and noise level. Each node is associated with a p-dimensional feature vector, and a subset of nodes is designated as *informative*, meaning their features influence the graph-level binary target. The remaining *non-informative* nodes serve as noise. For each sample, we generate five parallel graphs—one for each data modality—with different noise structures. Each

modality yields a binary graph-level label, and a shared target is defined by aggregating these modality-specific labels to enable multimodal classification.

We conduct three experiments. In **Experiment 1**, the features of informative nodes are drawn from a modality-specific multivariate Gaussian distribution with correlated entries, and labels are assigned using a linear thresholding rule (see *Setting 1* in Appendix A7 for more details). Graphs contain 5 nodes (all informative), and the sample size is 100. In **Experiment 2**, the features for informative nodes are generated using a Gaussian Process to induce temporal structure across features. Labels are computed using a nonlinear function involving sinusoidal and quadratic terms (see *Setting 2* in Appendix A7 for more details). Graphs again contain 5 informative nodes, and the sample size is 100. **Experiment 3** follows the same setting as Experiment 2 but increases the graph size and sample size. Each graph has 50 nodes, with 40 designated as informative. The sample size is increased to 2,000, allowing us to assess MGMT's performance at scale under complex, multimodal conditions.

According to Figure 2, across all experiments, MGMT consistently outperforms feature concatenation and multimodal fusion baselines, with the most notable gains observed in the large-scale setting. TableA3 shows accuracy degrades when adaptive depth, supernode filtering, or inter-modality edges are removed, and degrades most when both the meta-graph and adaptive depth are disabled; confirming the importance of hierarchical graph reasoning and dynamic layer aggregation.

4.5 Neuroscience Applications

4.5.1 Local field potential (LFP) activity dataset

We apply our method to a challenging neuroscience problem: predicting the stimulus presented on a given trial using only LFP activity from the hippocampus. In this experiment [31, 1], subjects (rats) received repeated presentations of a sequence of stimuli (odors ABCDE) at a single odor port and were required to accurately identify each stimulus as being presented in the correct (e.g., AB<u>C</u>...) or incorrect sequence position (e.g., AB<u>D</u>...) to receive a reward. Neural activity, including both spiking and LFP activity, was recorded from the dorsal CA1 subregion of the hippocampus as they performed the task. Here we focus on the LFP activity data from the 5 subjects (SuperChris, Barat, Stella, Mitt, and Buchanan), collected from 20 to 22 electrodes (which varied between subjects), and sampled at 1,000 Hz. We treated each rat as a distinct "modality" and applied our proposed MGMT framework to borrow power across subjects in order to improve the overall decoding of LFP signals.

Each trial is associated with one shared stimulus label (A,B,C,D or E), and we construct a separate graph for each rat per trial using its own electrode-level LFP signals. Nodes represent electrodes (which vary in number and identity across subjects), and edges capture intra-subject correlations. We then build a meta-graph by linking "supernodes" across rats when their latent embeddings are similar under MGMT's localized attention as an operation justified by the common graph-signal smoothness prior (i.e., nearby nodes in the latent space tend to express similar activity patterns). Crucially, Superedges are aligning comparable brain dynamics across animals, effectively "borrowing statistical strength" across rats to reduce noise, and stabilize the trial-level representation used for decoding. This is not meant to just simply connect various brain regions across rats, rather alignment of their brain dynamics to strengthen the overall signals by properly borrowing power across rats.

As shown in Table A2, MGMT achieves the highest accuracy (**42.1%** $\pm$ 0.0252) predicting which odor (A–E) was presented on each trial using the LFP dataset, outperforming all baseline and fusion models, including the second-best model MT (39.2%) and other strong multimodal baselines such as MMGL (39.28%), MultiMoDN (37.8%), and FlexCare (36.4%). Traditional concatenation-based approaches like DNN and GNN yield substantially lower performance (30.6% and 27.8%, respectively), highlighting the difficulty of this cross-rat decoding task. For context, the theoretical chance level for this five-class problem is 20%, so MGMT exceeds chance by roughly $2.1\times$. To our knowledge, these results provide the first direct evidence that the stimulus presented on a given trial can be accurately predicted based on hippocampal LFP activity alone, which highlights the potential of graph data integration approaches in general and the potential of the MGMT model specifically.

Ablation results (Table A3) confirm that each architectural component contributes meaningfully to MGMT's performance, with the full model achieving the highest accuracy across all datasets.

Results of interpretation component. From a neuroscience perspective, the main results of the interpretation model based on MGMT's interpretation performance on LFP dataset (Fig. 3) are as

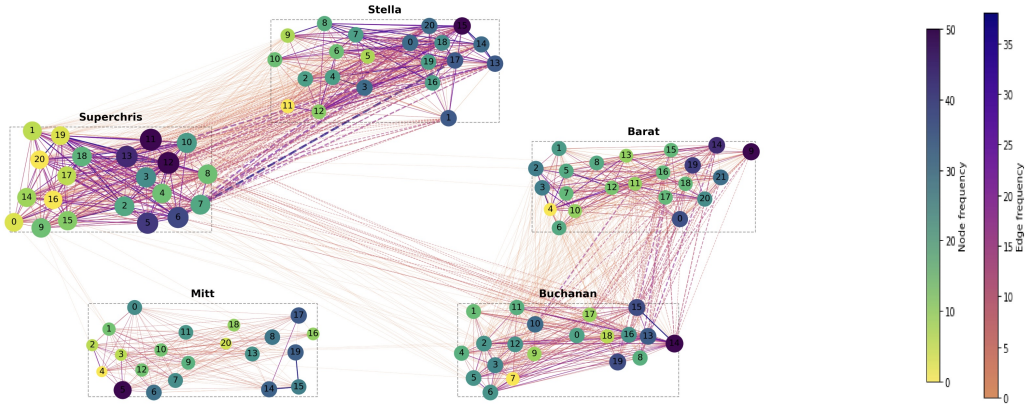

Figure 3: Cross-animal supernode and edge frequency map generated by the MGMT model. Each dashed box corresponds to one rat (Superchris, Stella, Barat, Mitt, Buchanan), with node size and color indicating the frequency of supernode selection across trials. Solid lines within each box represent within-rat edges, while dashed lines across boxes denote cross-rat superedges; line color and width reflect edge occurrence frequency. High-frequency supernodes and edges are concentrated in distal CA1 regions (right side), with cross-rat superedges predominantly linking distal regions across animals, while Mitt shows weaker connectivity patterns.

follows. First, we found that informative electrodes clustered on the right side of the electrode array. More specifically, we found that the highest-frequency supernodes and the strongest within-subject connections were consistently concentrated on that side, and that the pattern was consistent across subjects. This specific clustering makes sense given that the two electrode arrays targeted different segments of the CA1 region: electrodes on the right targeted the distal segment, electrodes on the left the proximal region. The distal segment, where most informative electrodes are located, is more strongly associated with non-spatial inputs (e.g., odors, objects) and the proximal segment with visuospatial inputs. Such clustering of informative electrodes in distal CA1 is also consistent with previous work focusing on a different type of non-spatial trial classification (in sequence vs out of sequence [32]). Second, there were interesting variations in the pattern of informative edges across subjects. Although they showed a similar pattern of informative nodes, some subjects showed weaker relationships in edges. For example, one subject (Mitt) showed fewer strong within-subject edges and lower-frequency superedges. We also found that the pattern of superedges detected strong relationships between pairs of subjects (Stella-SuperChris, SuperChris-Buchanan, Buchanan-Barat), which did not extend to all subjects involved (e.g., weak Stella-Barat relationship). It remains unclear what aspect of the signal produced such edge variation, but possible interpretations include variability in electrode locations (e.g., depth relative to the cell layer), noise levels, or subjects' task performance. In sum, the interpretation model provided the necessary neuroscience framework to identify the key aspects of the LFP signal that supported classification accuracy and offered novel insights into potential mechanisms to examine in future work.

5 Conclusion, Limitations, and Future Work

We introduced the Multi-Graph Meta-Transformer (MGMT), a unified framework for structured multi-graph learning that combines graph-specific Graph Transformer encoders with a meta-graph over learned supernodes and superedges, plus an adaptive depth-aware fusion to aggregate hierarchical representations. Across synthetic and neuroscience datasets, MGMT improves both accuracy and interpretability over standard fusion baselines.

Limitations include (i) reliance on thresholded similarity for meta-graph edges, (ii) rising compute with more modalities and larger graphs due to attention layers, and (iii) attention-based importance scores that may not capture causal structure in noisy, high-dimensional settings.

Future work will explore learnable edge weighting in place of thresholding, sparse/low-rank attention for scalability, causal attribution and counterfactual analyses for deeper interpretability, extensions to dynamic/temporal graphs, and pretraining to improve generalization.

References

[1] Babak Shahbaba, Lingge Li, Forest Agostinelli, Mansi Saraf, Keiland W Cooper, Derenik Haghverdian, Gabriel A Elias, Pierre Baldi, and Norbert J Fortin. Hippocampal ensembles represent sequential relationships among an extended sequence of nonspatial events. *Nature communications*, 13(1):787, 2022.

[2] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[3] Yanfu Zhang, Hongchang Gao, Jian Pei, and Heng Huang. Robust self-supervised structural graph neural network for social network prediction. In *Proceedings of the ACM Web Conference 2022*, pages 1352–1361, 2022.

[4] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

[5] Weizhi Xu, Junfei Wu, Qiang Liu, Shu Wu, and Liang Wang. Evidence-aware fake news detection with graph neural networks. In *Proceedings of the ACM web conference 2022*, pages 2501–2510, 2022.

[6] Zhixun Li, Dingshuo Chen, Qiang Liu, and Shu Wu. The devil is in the conflict: Disentangled information graph neural networks for fraud detection. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1059–1064. IEEE, 2022.

[7] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[8] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[9] Xingtong Yu, Zemin Liu, Yuan Fang, and Xinming Zhang. Learning to count isomorphisms with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4845–4853, 2023.

[10] Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

[11] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[12] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[13] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *Advances in Neural Information Processing Systems*, 35:14582–14595, 2022.

[14] Yufei He, Yuan Sui, Xiaoxin He, Yue Liu, Yifei Sun, and Bryan Hooi. Unigraph2: Learning a unified embedding space to bind multimodal graphs. In *Proceedings of the ACM on Web Conference 2025*, pages 1759–1770, 2025.

[15] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803, 2019.

[16] Nasir Hayat, Krzysztof J Geras, and Farah E Shamout. Medfuse: Multi-modal fusion with clinical time-series data and chest x-ray images. In *Machine Learning for Healthcare Conference*, pages 479–503. PMLR, 2022.

[17] Muhao Xu, Zhenfeng Zhu, Youru Li, Shuai Zheng, Yawei Zhao, Kunlun He, and Yao Zhao. Flexcare: Leveraging cross-task synergy for flexible multimodal healthcare prediction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3610–3620, 2024.

[18] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[19] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[20] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[21] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[22] Shuai Zheng, Zhenfeng Zhu, Zhizhe Liu, Zhenyu Guo, Yang Liu, Yuchen Yang, and Yao Zhao. Multi-modal graph learning for disease prediction. *IEEE Transactions on Medical Imaging*, 41(9):2207–2216, 2022.

[23] Vinitra Swamy, Malika Satayeva, Jibril Frej, Thierry Bossy, Thijs Vogels, Martin Jaggi, Tanja Käser, and Mary-Anne Hartley. Multimodn—multimodal, multi-task, interpretable modular networks. *Advances in neural information processing systems*, 36:28115–28138, 2023.

[24] Mengmeng Ma, Jian Ren, Long Zhao, Davide Testuggine, and Xi Peng. Are multimodal transformers robust to missing modality? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18177–18186, 2022.

[25] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19314–19326. Curran Associates, Inc., 2020.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[27] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[28] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, Andrew Y Ng, et al. Multimodal deep learning. In *ICML*, volume 11, pages 689–696, 2011.

[29] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.

[30] Billy Pik Lik Lau, Sumudu Hasala Marakkalage, Yuren Zhou, Naveed Ul Hassan, Chau Yuen, Meng Zhang, and U-Xuan Tan. A survey of data fusion in smart city applications. *Information Fusion*, 52:357–374, 2019.

[31] Timothy A Allen, Daniel M Salz, Sam McKenzie, and Norbert J Fortin. Nonspatial sequence coding in ca1 neurons. *Journal of Neuroscience*, 36(5):1547–1563, 2016.

[32] Wenzhuo Zhou, Annie Qu, Keiland W Cooper, Norbert Fortin, and Babak Shahbaba. A model-agnostic graph neural network for integrating local and global information. *Journal of the American Statistical Association*, pages 1–14, 2024.

[33] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[34] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Hrayr Harutyunyan, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *The Thirty-sixth International Conference on Machine Learning (ICML)*, 2019.

[35] Yanteng Zhang, Xiaohai He, Yi Hao Chan, Qizhi Teng, and Jagath C. Rajapakse. Multimodal graph neural network for early diagnosis of alzheimer's disease from smri and pet scans. *Computers in Biology and Medicine*, 164:107328, 2023.

[36] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.

[37] Duane L. Beekly, Erin M. Ramos, Gerald van Belle, Woodrow Deitrich, Amber D. Clark, Mary E. Jacka, and Walter A. Kukull. The National Alzheimer's Coordinating Center (NACC) Database: an Alzheimer disease database. *Alzheimer Disease and Associated Disorders*, 18(4):270–277, December 2004.

[38] Sandra Weintraub, David P. Salmon, N. Mercaldo, Steven Ferris, Neill R. Graff-Radford, Helena Chui, Jeffrey L. Cummings, Charles, Decarli, Norman L. Foster, Douglas R. Galasko, Elaine R. Peskind, Woodrow Dietrich, Duane L. Beekly, Walter A. Kukull, C. John, and Morris. The Alzheimer's Disease Centers' Uniform Data Set (UDS): The Neuropsychological Test Battery. *Alzheimer Disease and Associated Disorders*, 23(2):91–101, 2009.

A1 Graph Transformer with Localized Graph-Aware Attention

The standard Transformer architecture employs a global self-attention mechanism in which every token attends to all others. This is computationally inefficient and often inappropriate in the context of graph-structured data, where meaningful interactions are localized to a node's immediate neighborhood. To bridge this gap, we adopt the localized graph-aware attention formulation proposed by Shi et al. [33], which restricts attention to a node's 1-hop neighbors.

To preserve self-information, we extend the neighborhood to include the node itself. Specifically, we define $\bar{\mathcal{N}}(u) = \mathcal{N}(u) \cup \{u\}$, ensuring each node can incorporate its own features during attention-based message passing.

Let $\boldsymbol{H}^{(l-1)} = \{\boldsymbol{H}_1^{(l-1)}, \ldots, \boldsymbol{H}_N^{(l-1)}\}$ denote the set of node features from the previous layer. Each node u aggregates information from its extended neighborhood $v \in \bar{\mathcal{N}}(u)$ using the following multi-head self-attention mechanism.

For each attention head $m = 1, \ldots, M$ and layer $\ell = 1, \ldots, L$:

1. Linear Projections (queries, keys, values):

$$\boldsymbol{Q}_u^{(l,m)} = \boldsymbol{W}_Q^{(l,m)} \boldsymbol{h}_u^{(l-1)} + \boldsymbol{b}_Q^{(l,m)}, \tag{A8}$$

$$\boldsymbol{K}_v^{(l,m)} = \boldsymbol{W}_K^{(l,m)} \boldsymbol{h}_v^{(l-1)} + \boldsymbol{b}_K^{(l,m)}, \tag{A9}$$

$$\boldsymbol{V}_v^{(l,m)} = \boldsymbol{W}_V^{(l,m)} \boldsymbol{h}_v^{(l-1)} + \boldsymbol{b}_V^{(l,m)}. \tag{A10}$$

The learnable matrices $\boldsymbol{W}_Q^{(l,m)}$, $\boldsymbol{W}_K^{(l,m)}$, and $\boldsymbol{W}_V^{(l,m)}$ are referred to as the *Query*, *Key*, and *Value* projection matrices, respectively. These matrices project each node's feature vector into three distinct spaces:

- The **Query** vector $\boldsymbol{Q}_u^{(l,m)}$ represents the type of information that node u seeks from its neighbors.
- The **Key** vector $\boldsymbol{K}_v^{(l,m)}$ encodes what information neighbor node v can provide.
- The **Value** vector $\boldsymbol{V}_v^{(l,m)}$ contains the actual content to be aggregated.

This separation allows the model to compute a relevance score between nodes before deciding how much information to share.

2. Attention Score Calculation: The attention coefficient from node u to neighbor $v \in \bar{\mathcal{N}}(u)$ is computed as:

$$\alpha_{uv}^{(l,m)} = \frac{\exp\left(\frac{\boldsymbol{Q}_u^{(l,m)\top} \boldsymbol{K}_v^{(l,m)}}{\sqrt{d_h}}\right)}{\sum_{r \in \mathcal{N}(u)} \exp\left(\frac{\boldsymbol{Q}_u^{(l,m)\top} \boldsymbol{K}_r^{(l,m)}}{\sqrt{d_h}}\right)}, \tag{A11}$$

where d_h is the dimensionality of each head.

3. Neighborhood Aggregation:

$$\boldsymbol{Z}_u^{(l,m)} = \sum_{v \in \bar{\mathcal{N}}(u)} \alpha_{uv}^{(l,m)} \boldsymbol{V}_v^{(l,m)}. \tag{A12}$$

4. Multi-Head Output and Update: The outputs from all heads are concatenated and linearly transformed:

$$\hat{\boldsymbol{H}}_u^{(l)} = \boldsymbol{W}_O^{(l)} \left[\boldsymbol{Z}_u^{(l,1)} \| \cdots \| \boldsymbol{Z}_u^{(l,M)}\right] + \boldsymbol{b}_O^{(l)}, \tag{A13}$$

where $\|$ denotes concatenation across heads, and $\boldsymbol{W}_O^{(l)} \in \mathbb{R}^{d \times d}$, $\boldsymbol{b}_O^{(l)} \in \mathbb{R}^d$ are learnable projections.

This formulation allows each node to dynamically attend to its extended local neighborhood, learning rich contextual representations while respecting the sparse structure of the input graph. The learned attention scores can also be used for interpretability and identifying important nodes and edges, as discussed in the main text.

A2 Depth-Aware Aggregation in MGMT

To enhance the robustness of graph-specific representation learning and mitigate sensitivity to the choice of Transformer depth, we introduce an adaptive depth-aware fusion strategy inspired by recent developments in graph learning [32]. Rather than relying on a fixed-depth stack, we aggregate node embeddings across multiple Transformer layers, weighted by their contribution to graph-level prediction performance.

Let $\boldsymbol{H}_{ik}^{(\ell)} \in \mathbb{R}^{N_i \times d}$ denote the node embeddings of graph i in instance k after the ℓ-th Graph Transformer layer, for $\ell = 1, \ldots, L$, $i = 1, \ldots, n$ and $k = 1, \ldots, K$. Here, K is the total number of samples (instances), and n is the number of graphs per instance. To evaluate the representational quality of each layer, we compute a graph-level representation by applying mean pooling over the node embeddings:

$$\bar{\boldsymbol{H}}_{ik}^{(\ell)} = \frac{1}{N_i} \mathbf{1}_{N_i}^{\top} \boldsymbol{H}_{ik}^{(\ell)} \in \mathbb{R}^{1 \times d}. \tag{A14}$$

Each pooled graph embedding $\bar{\boldsymbol{H}}_{ik}^{(\ell)}$ is passed through a lightweight classifier to obtain predictions, and its predictive quality is evaluated using the graph-level label. Let $Y_k \in \{1, \ldots, |\mathcal{Y}|\}$ be the true label for instance k. The classification error for graph i at depth ℓ is computed as:

$$\epsilon_i^{(\ell)} = \frac{\sum_{k=1}^{K} \beta_{ik}^{(\ell)} \mathbb{1}\left\{ Y_k \neq \arg\max_y \operatorname{softmax}\left(\bar{\boldsymbol{H}}_{ik}^{(\ell)} \right) \right\}}{\sum_{k=1}^{K} \beta_{ik}^{(\ell)}} \tag{A15}$$

where $\beta_{ik}^{(\ell)}$ is the weight assigned to graph i in instance k at depth ℓ.

The confidence score for the ℓ-th layer of graph i is defined as:

$$\Gamma_i^{(\ell)} = \frac{1}{2} \log \left(\frac{1 - \epsilon_i^{(\ell)}}{\epsilon_i^{(\ell)}} \right). \tag{A16}$$

To emphasize misclassified instances, sample weights are updated between depths using:

$$\beta_{ik}^{(\ell+1)} \propto \beta_{ik}^{(\ell)} \exp\left(\mathbb{1}\left\{ Y_k \neq \arg\max_Y \operatorname{softmax}\left(\bar{\boldsymbol{H}}_{ik}^{(\ell)} \right) \right\} \cdot \Gamma_i^{(\ell)} \right). \tag{A17}$$

The confidence scores $\Gamma_i^{(\ell)}$ are used to weight both the depth-wise fused node embeddings and the attention scores across Transformer layers, ensuring that layers contributing most to prediction are emphasized during super-node extraction and representation learning.

A3 Theoretical Properties

In this section, we establish MGMT's theoretical foundations through: (1) *intra-graph analysis*, demonstrating superior feature representation within individual graphs; and (2) *inter-graph analysis*, showing enhanced predictive power through meta-graph construction. Complete proofs appear in Appendix A4, with additional theoretical results in Appendix A5.

A3.1 Intra-graph analysis

We analyze the depth-aware mixing strategy in (3) which enables MGMT to aggregate information across different depths of message passing. First, we establish some formal definitions.

Let $\mathcal{M}(\boldsymbol{A}) \in \mathbb{R}^{N \times N}$ be a message passing operator on an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, e.g., the augmented adjacency matrix, $\mathcal{M}(\boldsymbol{A}) = \boldsymbol{A} + \boldsymbol{I}$. Given $\mathcal{M}(\boldsymbol{A})$ and an activation function σ, denote the 1-hop feature aggregation as

$$\mathcal{U}(\boldsymbol{X}; \mathcal{M}(\boldsymbol{A}), \sigma) \coloneqq \sigma(\mathcal{M}(\boldsymbol{A})\boldsymbol{X}),$$

and the ℓ-hop aggregation is the ℓ-fold composition of $\mathcal{U}$, namely,

$$\mathcal{U}^{\ell}(\boldsymbol{X}; \mathcal{M}(\boldsymbol{A}), \sigma) \coloneqq \underbrace{\sigma(\mathcal{M}(\boldsymbol{A}) \cdots \sigma(\mathcal{M}(\boldsymbol{A})}_{\ell \text{ times}} \boldsymbol{X})).$$

Building on these definitions, we introduce L**-hop mixing**, which characterizes a model's ability to represent multi-depth information. While originally studied for Graph Convolutional Networks with graph Laplacians [34, 32], we extend this concept to general message passing operators.

Definition A1 (L-hop mixing with general message passing)**.** *Given $\mathcal{M}(\cdot)$, a model is capable of representing L-hop mixing if for any $\eta_1, \ldots, \eta_L \in \mathbb{R}$, there exists a setting of its parameter and an injective (one-to-one) mapping $f(\cdot)$, such that the output of the model is equivalent as*

$$f\left(\sum_{\ell=1}^{L} \eta_\ell \cdot \mathcal{U}^\ell(\boldsymbol{X}; \mathcal{M}(\boldsymbol{A}), \sigma)\right), \tag{A18}$$

for any adjacency matrix $\boldsymbol{A}$, activation function σ, and node features $\boldsymbol{X}$.

Remark A2. *If $\mathcal{M}(\boldsymbol{A}) = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{D}$ is the diagonal degree matrix with $D_{ii} = \sum_{j=1}^{N} A_{ij} + 1$, Definition A1 recovers the L-hop mixing with Graph Laplacian in the GCN literature [34, 32].*

Our first theoretical result demonstrates that depth-aware Graph Transformers in MGMT can represent L-hop mixing for each graph.

Theorem A3. *With message passing operator $\mathcal{M}(\boldsymbol{A}) = \text{softmax}(\boldsymbol{A} + \boldsymbol{I})$, where* softmax *is applied row-wise. MGMT's depth-aware Graph Transformers in* (1)–(3) *can represent L-hop mixing.*

The proof appears in Appendix A4. Notably, we also demonstrate in Appendix A5.1 that vanilla Graph Transformers **cannot** learn L-hop neighborhood mixing.

A3.2 Inter-graph analysis

This section analyzes how MGMT's meta-graph construction boosts prediction power compared to late fusion approaches [35].

Recall from Section 2.1.3, the meta-graph $\mathcal{G}_M = (\mathcal{S}_M, \mathcal{E}_M)$ combines super-nodes $\mathcal{S}_M = \bigcup_{i=1}^{n} \mathcal{S}_i$. Its initial embedding $\boldsymbol{H}_M^{(0)} \in \mathbb{R}^{|\mathcal{S}_M| \times d}$ stacks super-node embeddings where $\forall u \in \mathcal{S}_i$, $\boldsymbol{H}_{M,u}^{(0)} = \boldsymbol{H}_{i,u}$. MGMT applies additional L_{GT} Graph Transformer layers followed by a global pooling to obtain the final graph-level embedding. Lastly, we apply L_{MLP} MLP layers for class probabilities. Assume without of loss of generality that $L_{\text{GT}} = 1$ and $L_{\text{MLP}} = 2$, the function class of MGMT given $\boldsymbol{H}_M^{(0)}$ can be expressed as

$$\mathcal{F}_M = \left\{ f : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{|\mathcal{Y}|} \;\middle|\; f = \boldsymbol{W}_{\text{MLP}}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP}}^{(1)} \text{Pool}(\text{GT}(\boldsymbol{H}_M^{(0)}))\Big) \right\}, \tag{A19}$$

where $\text{GT}(\cdot) : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{|\mathcal{S}_M| \times d}$ is the Graph Transformer, $\text{Pool}(\cdot) : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{h'}$ is a graph pooling, and $\boldsymbol{W}_{\text{MLP}}^{(1)} \in \mathbb{R}^{h' \times h''}$, $\boldsymbol{W}_{\text{MLP}}^{(2)} \in \mathbb{R}^{|\mathcal{Y}| \times h''}$ are MLP weight matrices, with $h', h'' \in \mathbb{N}^+$. All subsequent analysis could be easily extended to any number of L_{MLP} and L_{GT}.

We consider the late fusion strategy that employs weighted averaging of class probabilities from graph-specific models. Formally, the late fusion classification function can be represented as

$$\mathcal{F}_{\text{late}} = \left\{ f : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{|\mathcal{Y}|} \;\middle|\; f = \sum_{i=1}^{n} w_i \cdot \boldsymbol{W}_{\text{MLP},i}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP},i}^{(1)} \text{Pool}_{\mathcal{S}_i}(\boldsymbol{H}_M^{(0)})\Big) \right\},$$

where $\{\boldsymbol{W}_{\text{MLP},i}^{(l)}\}_{l \in [2], i \in [n]}$ is the set of graph-specific MLP parameter, and the set of late fusion weights is $\{w_i \in \mathbb{R}\}_{i \in [n]}$ such that $\sum_{i=1}^{n} w_i = 1$.

Given the joint distribution of a feature-label pair $(\boldsymbol{X}, Y) \sim \mathcal{P}$ and a loss function $\mathcal{L}$, denote the generalization error of a function f as

$$R(f; \mathcal{P}, \mathcal{L}) \coloneqq \mathbb{E}_{(\boldsymbol{X},Y) \sim \mathcal{P}}[\mathcal{L}(f(\boldsymbol{X}), Y)]$$

Following [36], we define the **approximation error** of a function class $\mathcal{F}$ as the minimum generalization error achievable by a function in $\mathcal{F}$, namely,

$$\epsilon(\mathcal{F}; \mathcal{P}, \mathcal{L}) \coloneqq \inf_{f \in \mathcal{F}} R(f; \mathcal{P}, \mathcal{L}). \tag{A20}$$

Assume latent representations of the meta graph follow $(\boldsymbol{H}_M^{(0)}, Y) \sim \mathcal{P}_M$. The next theorem shows MGMT is a more powerful graph fusion framework compared to late fusion in the sense that it achieves smaller approximation error.

Theorem A4. *Denote approximation error of MGMT on the meta-graph as $\epsilon(\mathcal{F}_M; \mathcal{P}_M, \mathcal{L})$, and the approximation error of late fusion of graph-specific classifiers $\epsilon(\mathcal{F}_{\text{late}}; \mathcal{P}_M, \mathcal{L})$, then*

$$\epsilon(\mathcal{F}_M; \mathcal{P}_M, \mathcal{L}) \leq \epsilon(\mathcal{F}_{\text{late}}; \mathcal{P}_M, \mathcal{L}).$$

The proof appears in Appendix A5. We also demonstrate MGMT outperforms another popular graph fusion alternative — late fusion, in Appendix A5.2.

A4 Mathematical Proofs

Proof of Theorem A3. For simplicity, we omit graph-specific subscripts throughout the proof (e.g. $\boldsymbol{X}$ instead of $\boldsymbol{X}_i$) as the arguments apply universally for all graphs. Consider the Graph Transformer (GT) structure with a single head $m = 1$. For each layer $\ell = 1, \ldots, L$, let $\boldsymbol{W}_Q^{(\ell)} = \boldsymbol{W}_K^{(\ell)} = \boldsymbol{0}$, $\boldsymbol{W}_V^{(\ell)} = \boldsymbol{I}$, and $\boldsymbol{b}_V^{(\ell)} = \boldsymbol{0}$ in (1). Here $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{0}$ denotes matrix/vector of all zeros. For the feedforward layer in (2), set weights as $\boldsymbol{I}$, bias as $\boldsymbol{0}$, and remove the residual connection and normalization layer. Then for each edge $(u, v) \in \mathcal{E} \cup \{(u, u)\}$, the updating rules in (1) and (2) simplifies to

$$\begin{aligned}
\boldsymbol{Q}_u^{(\ell)} &= \boldsymbol{b}_Q^{(\ell)}, \\
\boldsymbol{K}_v^{(\ell)} &= \boldsymbol{b}_K^{(\ell)}, \\
\boldsymbol{V}_v^{(\ell)} &= \boldsymbol{H}_v^{(\ell-1)}, \\
\alpha_{uv}^{(\ell)} &= \frac{\exp\left(\frac{\boldsymbol{Q}_u^{(\ell)\top} \boldsymbol{K}_v^{(\ell)}}{\sqrt{d}}\right)}{\sum_{v' \in \bar{\mathcal{N}}(u)} \exp\left(\frac{\boldsymbol{Q}_u^{(\ell)\top} \boldsymbol{K}_{v'}^{(\ell)}}{\sqrt{d}}\right)}, \\
\boldsymbol{H}_u^{(\ell)} &= \sigma\left(\sum_{v \in \bar{\mathcal{N}}(u)} \alpha_{uv}^{(\ell)} \boldsymbol{V}_v^{(\ell)}\right).
\end{aligned}$$

It is clear that the attention matrix $\boldsymbol{\alpha}^{(\ell)}$ reduces to $\mathcal{M}(\boldsymbol{A}) = \text{softmax}(\boldsymbol{A} + \boldsymbol{I})$. Recall that the initial embedding $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, we can explicitly expand the recursive updating rule above, and write the embeddings for each layer ℓ in the following compact form:

$$\boldsymbol{H}^{(\ell)} = \mathcal{U}^{\ell}(\boldsymbol{X}; \mathcal{M}(\boldsymbol{A}), \sigma).$$

Let $\Gamma^{(\ell)} = \eta_\ell$, for $\ell = 1, \ldots, L$ in (3), the graph-specific fused embeddings can be represented as

$$\sum_{\ell=1}^{L} \eta_\ell \cdot \mathcal{U}^{\ell}(\boldsymbol{X}; \mathcal{M}(\boldsymbol{A}), \sigma),$$

which satisfies Definition A1 with identity mapping $f(\cdot)$. □

Remark A1. *While the depth-aware fusion step in* (3) *is highly flexible and can accommodate any set of weights $\{\Gamma_\ell\}_{\ell=1}^{L}$, we employ the confidence score weights defined in equation Appendix A2 to adaptively aggregate the latent representations that yield the highest classification accuracy.*

Proof of Theorem A4. Similar to the proof of Theorem A2, we will show $\mathcal{F}_{\text{late}} \subseteq \mathcal{F}_M$ and the desired results follows directly from the definition of approximation error in (A20).

Consider a class of pooling function that concatenates the graph-specific pooled embeddings, formally,

$$\text{ConcatPool}(\boldsymbol{H}_M^{(0)}) = \Big\|_{i=1}^{n} \text{Pool}_{\mathcal{S}_i}(\boldsymbol{H}_M^{(0)}), \tag{A21}$$

where $\|$ denotes the concatenation operation, $\text{Pool}_{\mathcal{S}_i}(\cdot) : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{h'}$, as defined in (A25), is the global pooling function restricted to $\mathcal{S}_i$. Hence $\text{ConcatPool}(\boldsymbol{H}_M^{(0)}) : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{nh'}$ represents the concatenation of graph-specific embeddings.

Further, let $D(\{\boldsymbol{W}_{\text{MLP},i}^{(1)}\}_{i=1}^n)$ be the diagonal block matrix with diagonal elements $\{\boldsymbol{W}_{\text{MLP},i}^{(1)}\}_{i=1}^n$, then one can easily check that (A21) can be rewritten as

$$\begin{aligned}\mathcal{F}_{\text{late}} = \Big\{ f : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{|\mathcal{Y}|} \;\Big|\; & f = \boldsymbol{W}_{\text{MLP}}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP}}^{(1)} \text{Pool}(\text{GT}(\boldsymbol{H}_M^{(0)}))\Big), \\ & \gamma > 1, \boldsymbol{W}_V = \boldsymbol{I}, \boldsymbol{b}_V = \boldsymbol{0}, \\ & \text{Pool}(\cdot) = \text{ConcatPool}(\cdot), \\ & \boldsymbol{W}_{\text{MLP}}^{(1)} = D(\{\boldsymbol{W}_{\text{MLP},i}^{(1)}\}_{i=1}^n), \\ & \boldsymbol{W}_{\text{MLP}}^{(2)} = w_1 \boldsymbol{W}_{\text{MLP},1}^{(2)} \| \cdots \| w_n \boldsymbol{W}_{\text{MLP},n}^{(2)} \Big\},\end{aligned} \tag{A22}$$

where $\gamma, \boldsymbol{W}_V, \boldsymbol{b}_V$ are parameters of the Graph Transformer layer as defined in (A26). Finally, from (A19) and (A22), it is clear that $\mathcal{F}_{\text{late}} \subseteq \mathcal{F}_M$, which concludes the proof. □

A5 Additional Theoretical Results

A5.1 Additional Intra-graph Results

Theorem A1. *Let $\mathcal{M}(\boldsymbol{A}) = \text{softmax}(\boldsymbol{A} + \boldsymbol{I})$ as in Theorem A3, the vanilla Graph Transformer is **not** capable of representing L-hop neighborhood mixing.*

Proof. Following a similar strategy in Abu-El-Haija et al. [34], it suffices to shows that the vanilla Graph Transformer (GT) fails to represent 2-hop mixing, which in turn implies the inability to represent the general L-hop mixing. Consider the particular case, where $m = 1$, $\sigma(x) = x$. As reviewed in Appendix A1, the final graph embedding of a vanilla GT with depth L can be represented as

$$\boldsymbol{H}^{(L)} = \left[\prod_{\ell=1}^{L} \text{softmax}\Big((\boldsymbol{A} + \boldsymbol{I}) \odot \boldsymbol{\alpha}^{(\ell)}\Big)\right] \boldsymbol{X} \prod_{\ell=1}^{L} \boldsymbol{W}_V^{(\ell)},$$

for attention matrices $\{\boldsymbol{\alpha}^{(\ell)}\}_{\ell=1}^L$ and weights $\{\boldsymbol{W}_V^{(\ell)}\}_{\ell=1}^L$. Here $\odot$ denote the Hadamard product. Let $\boldsymbol{W}^* = \prod_{\ell=1}^L \boldsymbol{W}_V^{(\ell)}$, and consider the case where $\eta_1 = 1$ and $\eta_2 = -1$. If the vanilla GT is able to represent 2-hop mixing, there exist an injective mapping f and a configuration of the parameters such that

$$\left[\prod_{\ell=1}^{L} \text{softmax}\Big((\boldsymbol{A} + \boldsymbol{I}) \odot \boldsymbol{\alpha}^{(\ell)}\Big)\right] \boldsymbol{X}\boldsymbol{W}^* = f(\mathcal{M}(\boldsymbol{A})\boldsymbol{X} - \mathcal{M}^2(\boldsymbol{A})\boldsymbol{X}) \tag{A23}$$

holds for any adjacency matrices $\boldsymbol{A}$ and node features $\boldsymbol{X}$.

Consider a fully disconnected graph with $\boldsymbol{A} = \boldsymbol{0}$ and $\boldsymbol{X}$, then $\mathcal{M}(\boldsymbol{A}) = \text{softmax}(\boldsymbol{I}) = \boldsymbol{I}$, and $\text{softmax}\big((\boldsymbol{A} + \boldsymbol{I}) \odot \boldsymbol{\alpha}^{(\ell)}\big) = \boldsymbol{I}$ for $\ell = 1, \ldots, L$, which implies $\boldsymbol{W}^* = f(\boldsymbol{0})$. On the other hand, consider a graph with a single edge between node 1 and 2, namely, $A_{12} = A_{21} = 1$ and 0 otherwise. Then

$$\mathcal{M}(\boldsymbol{A}) = \underbrace{\begin{bmatrix} 0.5 & 0.5 & 0 & \cdots & 0 \\ 0.5 & 0.5 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{:= \boldsymbol{A}^*}$$

Let $\boldsymbol{X} = \boldsymbol{A}^*$, then $f(\mathcal{M}(\boldsymbol{A})\boldsymbol{X} - \mathcal{M}^2(\boldsymbol{A})\boldsymbol{X}) = f(\boldsymbol{0})$. Furthermore, it is easy to check that

$$\prod_{\ell=1}^{L} \text{softmax}\Big((\boldsymbol{A} + \boldsymbol{I}) \odot \boldsymbol{\alpha}^{(\ell)}\Big) = \boldsymbol{A}^*,$$

since features of node 1 and 2 are identical. It follows that $\boldsymbol{A}^*\boldsymbol{W}^* = f(\boldsymbol{0})$.

Combining the two scenarios, we must have $(\boldsymbol{I} - \boldsymbol{A}^*)\boldsymbol{W}^* = \boldsymbol{0}$, which implies that $\boldsymbol{W}_1^* = \boldsymbol{W}_2^*$, where $\boldsymbol{W}_i^*$ is the i-th row of $\boldsymbol{W}^*$. Since the choice of node 1 and 2 was arbitrary, all rows of $\boldsymbol{W}^*$ should be identical, hence $\text{rank}(\boldsymbol{W}^*) \leq 1$ and $\text{rank}([\prod_{\ell=1}^{L} \text{softmax}\big((\boldsymbol{A} + \boldsymbol{I}) \odot \boldsymbol{\alpha}^{(\ell)}\big)]\boldsymbol{X}\boldsymbol{W}^*) \leq 1$, which means the output of f should be at most rank 1 matrices by the equivalence assumption in (A23). Hence, f cannot be injective which concludes the proof by contradiction. □

A5.2 Additional Inter-graph Results

Let $\boldsymbol{H}_{\mathcal{S}_i} = \{\boldsymbol{H}_{i,u}\}_{u \in \mathcal{S}_i}$ be the embeddings for super-nodes in $\mathcal{S}_i$. Single-graph classifiers that operates on $\boldsymbol{H}_{\mathcal{S}_i}$ can be expressed as

$$\mathcal{F}_i = \Big\{ f : \mathbb{R}^{|\mathcal{S}_i| \times d} \mapsto \mathbb{R}^{|\mathcal{Y}|} \;\Big|\; f = \boldsymbol{W}_{\text{MLP}}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP}}^{(1)} \text{Pool}(\boldsymbol{H}_{\mathcal{S}_i}) \Big) \Big\}. \tag{A24}$$

Assume latent representations of the meta graph follow $(\boldsymbol{H}_M^{(0)}, Y) \sim \mathcal{P}_M$, and $(\boldsymbol{H}_{\mathcal{S}_i}, Y) \sim \mathcal{P}_i$ where $\mathcal{P}_i$ is the marginal distribution of $\mathcal{P}_M$ restricted to $\mathcal{S}_i$. The next result shows MGMT achieves smaller approximation error by leveraging information across all graphs.

Proposition A2. *Denote approximation error of MGMT on the meta-graph as $\epsilon(\mathcal{F}_M; \mathcal{P}_M, \mathcal{L})$, and the approximation error of graph-specific classifiers on the sub-graph as $\epsilon(\mathcal{F}_i; \mathcal{P}_i, \mathcal{L})$, then*

$$\epsilon(\mathcal{F}_M; \mathcal{P}_M, \mathcal{L}) \leq \epsilon(\mathcal{F}_i; \mathcal{P}_i, \mathcal{L}).$$

Proof of Proposition A2. Without loss of generality, we focus on the cases where both MGMT and graph-specific classifiers has $L_{\text{MLP}} = 2$ layers of MLP and MGMT has $L_{\text{GT}} = 1$ layer of Graph Transformer as specified in (A19) and (A24). The same argument below applies to any number of L_{MLP} and L_{GT}.

First, consider the function class that operates on the meta-graph but only utilizes the nodes from graph i, namely,

$$\bar{\mathcal{F}}_i = \Big\{ f : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{|\mathcal{Y}|} \;\Big|\; f = \boldsymbol{W}_{\text{MLP}}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP}}^{(1)} \text{Pool}_{\mathcal{S}_i}(\boldsymbol{H}_M^{(0)}) \Big) \Big\}, \tag{A25}$$

where $\text{Pool}_{\mathcal{S}_i}$ denote the global pooling operation that restricts on the nodes in $\mathcal{S}_i$. Since

$$\text{Pool}_{\mathcal{S}_i}(\boldsymbol{H}_M^{(0)}) = \text{Pool}(\boldsymbol{H}_{\mathcal{S}_i}),$$

we have that

$$R\Big(\boldsymbol{W}_{\text{MLP}}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP}}^{(1)} \text{Pool}_{\mathcal{S}_i}(\boldsymbol{H}_M^{(0)}) \Big); \mathcal{P}_M, \mathcal{L} \Big) = R\Big(\boldsymbol{W}_{\text{MLP}}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP}}^{(1)} \text{Pool}(\boldsymbol{H}_{\mathcal{S}_i}) \Big); \mathcal{P}_i, \mathcal{L} \Big).$$

It follows that

$$\epsilon(\bar{\mathcal{F}}_i; \mathcal{P}_M, \mathcal{L}) = \epsilon(\mathcal{F}_i; \mathcal{P}_i, \mathcal{L}).$$

We claim that $\bar{\mathcal{F}}_i \subseteq \mathcal{F}_M$, and by definition of approximation error in (A20),

$$\epsilon(\mathcal{F}_M; \mathcal{P}_M, \mathcal{L}) \leq \epsilon(\bar{\mathcal{F}}_i; \mathcal{P}_M, \mathcal{L}) = \epsilon(\mathcal{F}_i; \mathcal{P}_i, \mathcal{L}).$$

It remains to show the function class inclusion. Note that we can rewrite $\bar{\mathcal{F}}_i$ as

$$\begin{aligned} \bar{\mathcal{F}}_i = \Big\{ f : \mathbb{R}^{|\mathcal{S}_M| \times d} \mapsto \mathbb{R}^{|\mathcal{Y}|} \;\Big|\; & f = \boldsymbol{W}_{\text{MLP}}^{(2)} \sigma\Big(\boldsymbol{W}_{\text{MLP}}^{(1)} \text{Pool}_{\mathcal{S}_i}(\text{GT}(\boldsymbol{H}_M^{(0)})) \Big), \\ & \gamma > 1, \boldsymbol{W}_V = \boldsymbol{I}, \boldsymbol{b}_V = \boldsymbol{0} \Big\}, \end{aligned} \tag{A26}$$

where γ is the threshold defined in Section 2.1.3 that determines the connectivity between nodes in the meta-graph, $\boldsymbol{W}_V, \boldsymbol{b}_V$ are parameters for values in the Graph Transformer layer. Setting $\gamma > 1$ results in a fully disconnected meta-graph and together with $\boldsymbol{W}_V = \boldsymbol{I}, \boldsymbol{b}_V = \boldsymbol{0}$, the Graph Transformer layer $\text{GT}(\cdot)$ reduces to an identity mapping, which establishes the equivalence in (A26).

Finally, from (A19) and (A26), it is clear that $\bar{\mathcal{F}}_i \subseteq \mathcal{F}_M$, which concludes the proof. □

Table A1: Model Category Summary with Fusion Strategy, Graph Modeling, and Attention Usage

Category	Model Type	Fusion Method	Novel Model	Graph Structured Modeling	Attention-Based
Single-Source (No Fusion)	Simple DNN	×	×	×	×
	Simple GNN	×	×	√	×
	Simple DiffPool	×	×	√	×
	Simple Transformer	×	×	×	√
	Simple Graph Transformer	×	×	√	√
Concatenation Fusion	Concatenated Features (DNN)	√	×	×	×
	Concatenated Features (GNN)	√	×	√	×
	Concatenated Features (DiffPool)	√	×	√	×
Multimodal Fusion Baselines	MMGL [22]	√	×	√	√
	MultiMoDN [23]	√	×	×	×
	MedFuse [16]	√	×	×	×
	FlexCare [17]	√	×	×	√
	Meta-Transformer (MT) [24]	√	×	×	√
MGMT Ablation Variants	MGMT w/o Adaptive Depth Selection	√	√	√	√
	MGMT w/o Supernode Selection	√	√	√	√
	MGMT w/o Inter-graph Edges	√	√	√	√
	MGMT w/o Intra-graph Edges	√	√	√	√
	MGMT w/o Meta-Graph and Adaptive Depth	√	√	√	√
Proposed Model	MGMT	√	√	√	√

A6 Detailed Descriptions of Baseline Models

This appendix details the baselines used to evaluate our method. Table A1 provides a summary comparison of the baseline models.

A6.1 Single-Source Models (No Fusion)

We assess per-source predictive signal with five baselines: (i) DNN on flattened node features (edges ignored); (ii) GNN (GCN) with message passing over the given topology; (iii) DiffPool for hierarchical pooling into coarser clusters [27]; (iv) Transformer over node-feature sequences (no structural encoding); and (v) Graph Transformer that attends over 1-hop neighborhoods to incorporate local structure.

A6.2 Feature-Concatenation Fusion Models

These models use early fusion: each source is encoded by a source-specific extractor, the resulting embeddings are concatenated, and a shared DNN classifier is applied. Concretely, we consider (i) DNN-fusion with per-source DNN encoders; (ii) GNN-fusion with per-source GCN layers and graph-level pooling prior to concatenation; and (iii) DiffPool-fusion using per-source DiffPool encoders to produce graph-level embeddings that are concatenated and classified by a DNN.

A6.3 Benchmark Fusion Models

We benchmark against recent multimodal frameworks with distinct fusion strategies: (i) MMGL [22], which learns shared/specific embeddings via modality-aware representation learning and models subject-level similarity with a GNN; (ii) MultiMoDN [23], a modular design with independent encoders and late fusion, without structural reasoning; (iii) MedFuse [16], which aligns modalities in a shared latent space using contrastive/reconstruction losses, without explicit intra- or inter-modality structure; (iv) FlexCare [17], which uses modality-specific encoders and a Transformer fusion layer for heterogeneous clinical data, but no graph-based reasoning; and (v) Meta-Transformer (MT) [24], which uses modality prompts with a shared Transformer over unstructured inputs, without topological modeling. MGMT differs by jointly capturing both intra- and inter-graph relations through an attention-based meta-graph.

Most of these benchmark models were not originally designed for graph-structured inputs (they expect tabular, imaging, or clinical features). To compare fairly, we first converted each graph into a fixed-length vector by running the same graph-specific encoder used in MGMT (TransformerConv with global pooling and adaptive-depth aggregation) and using the resulting graph-level embedding as a "tabular" feature vector. For methods with multi-stream inputs (e.g., MultiMoDN, FlexCare, MedFuse), we fed one embedding per graph; for single-stream methods (e.g., Meta-Transformer), we concatenated the graph embeddings. All baselines used identical train/val/test splits, per-graph

Table A2: Accuracy (± standard error) for different models across datasets.

Model	Alzheimer	LFP Data	Experiment 1	Experiment 2	Experiment 3
Concatenated Features (DNN)	62.1 ± 0.0091	30.6 ± 0.0228	61.87 ± 0.0227	56.10 ± 0.0113	63.74 ± 0.0056
Concatenated Features (GNN)	70.1 ± 0.0093	27.8 ± 0.0234	55.64 ± 0.0236	64.20 ± 0.0120	67.17 ± 0.0060
Concatenated Features (DiffPool)	69.4 ± 0.0070	31.5 ± 0.0176	53.78 ± 0.0175	65.80 ± 0.0089	71.81 ± 0.0044
MMGL	79.38 ± 0.0052	39.28± 0.0193	59.20 ± 0.0104	62.80 ± 0.0084	68.75 ± 0.0012
MultiMoDN	76.4 ± 0.0075	37.8 ± 0.0182	60.40 ± 0.0167	61.50 ± 0.0101	65.10 ± 0.0050
MedFuse	75.2 ± 0.0084	35.1 ± 0.0171	59.70 ± 0.0152	64.35 ± 0.0096	63.84 ± 0.0053
FlexCare	76.14 ± 0.0079	36.4 ± 0.0188	61.10 ± 0.0139	69.82 ± 0.0091	64.03 ± 0.0056
MT	81.29 ±0.0092	39.20 ± 0.0296	62.31 ± 0.0124	66.30 ± 0.0112	69.24 ± 0.0034
MGMT	83.1 ± 0.0084	42.1 ± 0.0252	65.47 ± 0.0239	69.90 ± 0.0119	73.21 ± 0.0059

standardization, a learned linear projection to align embedding dimensions when required, and the same Optuna budget for hyperparameter tuning.

A6.4 Ablation Study

We assess the contribution of MGMT components by altering one module at a time while keeping the rest fixed: (i) w/o Adaptive Depth Selection: replace confidence-weighted layer aggregation with final-layer only, disabling depth-wise ensembling; (ii) w/o Supernode Selection: bypass attention-based node filtering so all nodes enter the meta-graph, increasing size and noise; (iii) w/o Inter-graph Edges: keep only within-graph edges to remove cross-graph interactions; (iv) w/o Intra-graph Edges: keep only cross-graph edges, removing within-graph structure; (v) w/o Meta-Graph and Adaptive Depth: omit the meta-graph, fix encoder depth, and perform late fusion via concatenated pooled graph outputs.

A7 Details on Simulation Settings

This section provides detailed descriptions of the synthetic data generation processes used in our simulation studies. We consider two controlled settings designed to evaluate the performance of MGMT under varying conditions of noise, feature dependency, and label complexity. Below, we describe the procedures for *Setting 1*, which uses modality-specific noise and a linear classification rule, and *Setting 2*, which introduces temporal dependencies and nonlinear label generation.

Setting 1: Feature Generation with Modality-Specific Noise and Linear Classification Rule

Let each graph consist of N nodes and d features per node. Define a subset of informative nodes $V_0 \subset \{1, \ldots, N\}$ with $|V_0| = N_0 < N$, and let $V_1 = \{1, \ldots, N\} \setminus V_0$ denote the non-informative nodes.

For each modality $i = 1, \ldots, n$, with modality-specific noise level σ_i, and for each graph sample $k = 1, \ldots, K$, node features are generated as follows:

- informative nodes $j \in V_0$ have features $\boldsymbol{x}_j^{(k,i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_i)$, where $\mathbf{\Sigma}_i \in \mathbb{R}^{d \times d}$ has ones on the diagonal and off-diagonal entries sampled uniformly from $[-\sigma_i, \sigma_i]$.
- Non-informative nodes $j \in V_1$ have features $\boldsymbol{x}_j^{(k,i)} \sim \text{Unif}(0, 0.5)^d$.

The modality-specific graph-level binary label $y_i^{(k)} \in \{0, 1\}$ is determined by the features of informative nodes:

$$y_i^{(k)} = \mathbb{I}\left(\frac{1}{|V_0|} \sum_{j \in V_0} \sum_{r=1}^{d} x_{j,r}^{(k,i)} + \varepsilon^{(k)} > 0 \right), \quad \varepsilon^{(k)} \sim \mathcal{N}(0, 0.1).$$

To enable multimodal fusion, a shared target variable is defined by aggregating modality-specific labels:

$$y_{\text{shared}}^{(k)} = \mathbb{I}\left(\sum_{i=1}^{n} w_i y_i^{(k)} \geq \tau \right),$$

where $w_i \in [0, 1]$ are modality weights summing to one, and $\tau \in [0, 1]$ is a threshold parameter.

Table A3: Accuracy (± standard error) for different ablation models across datasets.

Model	Alzheimer	LFP Data	Experiment 1	Experiment 2	Experiment 3
MGMT w/o Adaptive Depth Selection	81.2 ± 0.0085	40.6 ± 0.0223	64.20 ± 0.0240	68.80 ± 0.0117	71.45 ± 0.0057
MGMT w/o Supernode Selection	78.2 ± 0.0087	41.0 ± 0.0219	62.11 ± 0.0216	67.3 ± 0.0107	69.31 ± 0.0053
MGMT w/o Inter-graph Edges	76.5 ± 0.0088	38.9 ± 0.0214	61.72 ± 0.0225	66.90 ± 0.0121	68.35 ± 0.0051
MGMT w/o Intra-graph Edges	32.4 ± 0.0243	39.0 ± 0.0097	63.09 ± 0.0242	66.6 ± 0.0123	66.75 ± 0.0062
MGMT w/o Meta-Graph and Adaptive Depth	70.1 ± 0.0093	27.8 ± 0.0234	55.64 ± 0.0236	64.20 ± 0.0120	67.17 ± 0.0060
MGMT	83.1 ± 0.0084	42.1 ± 0.0252	65.47 ± 0.0239	69.90 ± 0.0119	73.21 ± 0.0059

Setting 2: Temporal Feature Dependency via Gaussian Process

In this setting, features of informative nodes are generated using a Gaussian Process (GP) to introduce temporal dependency across the d features. For $t = 1, \ldots, d$, let $x_t \sim \text{Unif}(0, 1)$, and define the GP with zero mean and a squared exponential kernel:

$$k(x_t, x_{t'}) = \sigma^2 \exp\left(-\frac{(x_t - x_{t'})^2}{l^2}\right),$$

with length-scale $l = 1$ and variance $\sigma^2 = 1$.

For non-informative nodes, features are also sampled from a GP with the same mean function, but with increased kernel variance $\sigma^2 = 2.5$, thereby injecting greater noise and reducing relevance for the target prediction.

The binary target label is defined using a nonlinear and complex function of the averaged features across informative nodes. Let

$$\boldsymbol{x} = \frac{1}{|V_0|} \sum_{j \in V_0} \boldsymbol{x}_j \in \mathbb{R}^d,$$

and define three projection vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3 \in \mathbb{R}^d$, each selecting a distinct third of the features:

$$\begin{aligned}
\boldsymbol{e}_1 &= [\underbrace{1, \ldots, 1}_{d/3}, \underbrace{0, \ldots, 0}_{2d/3}], \\
\boldsymbol{e}_2 &= [\underbrace{0, \ldots, 0}_{d/3}, \underbrace{1, \ldots, 1}_{d/3}, \underbrace{0, \ldots, 0}_{d/3}], \\
\boldsymbol{e}_3 &= [\underbrace{0, \ldots, 0}_{2d/3}, \underbrace{1, \ldots, 1}_{d/3}].
\end{aligned}$$

The graph-level label is then computed as:

$$y = \mathbb{I}\left(\sin(\boldsymbol{x}^\top \boldsymbol{e}_1) \cdot \cos(\boldsymbol{x}^\top \boldsymbol{e}_2) + (\boldsymbol{x}^{\circ 2})^\top \boldsymbol{e}_3 + \varepsilon > 0\right), \quad \varepsilon \sim \mathcal{N}(0, 0.1),$$

where $\boldsymbol{x}^{\circ 2}$ denotes the element-wise square of $\boldsymbol{x}$, i.e., the Hadamard power.

Software implementing the algorithms and data experiments are available online at : `https://anonymous.4open.science/r/new_submission-33A6`

A8 Alzheimer Dataset

To demonstrate MGMT's generalizability beyond LFP data analysis, we have also applied it to an Alzheimer's disease (AD) detection problem as an example of broader biomedical applications. More specifically, we apply our method to the data obtained from the National Alzheimer's Coordinating Center (NACC), which standardizes data collected across 46 Alzheimer's Disease Research Centers (ADRCs) in the United States [37, 38]. The cohort comprises 1,237 subjects (61.5% HC and 38.5% MCI/AD) with both clinical assessments from the Uniform Data Set (UDS) and structural MRI available. Our goal is to separate subjects with mild cognitive impairment (MCI) or dementia due to Alzheimer's disease from healthy controls (HC).

Following our terminology, a setting is *multi-modal* when each subject is measured via distinct data sources (e.g., MRI vs. clinical assessments) that inhabit different feature spaces and sensing

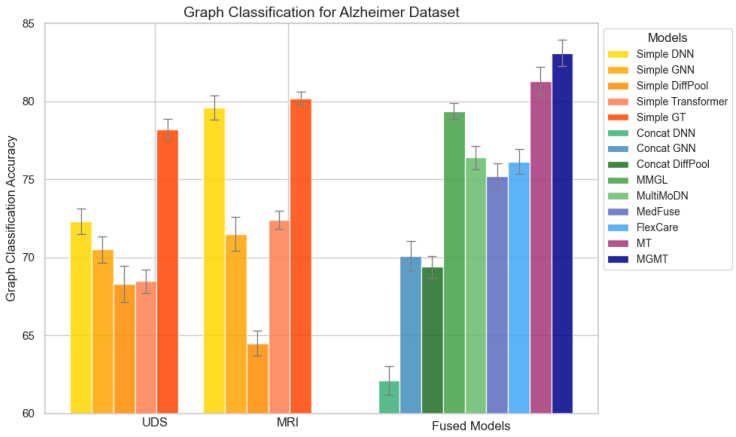

Figure A4: Test accuracies of single-source and fusion models Alzheimer's disease data. Each bar represents the average test accuracy across 5 folds, along with the corresponding standard error. MGMT consistently outperforms all other models, demonstrating the advantage of modeling intra- and inter-graph interactions.

processes. As shown in Figure A4, the MGMT model consistently outperformed both single-source and baseline fusion models. This highlights the importance of structure-aware joint fusion in multimodal biomedical prediction tasks. Moveover, ablations in figure A5 show that intra-graph structure and the meta-graph are critical: removing intra-graph edges collapses performance (32.4% vs. 83.1%), removing the meta-graph lowers accuracy to 70.1%, while dropping inter-graph edges (76.5%), supernode selection (78.2%), or adaptive depth (81.2%) yields progressively smaller but consistent declines.

A9 Experimental Setting and Efficiency Analysis

We evaluate the computational complexity and efficiency of MGMT through both theoretical and empirical analysis. This section is structured as follows: Section A9.1 presents a theoretical runtime complexity analysis of MGMT's core components; Section A9.2 provides empirical scalability results across four key input dimensions; Section A9.3 offers runtime profiling and efficiency comparisons, including infrastructure details and training costs.

A9.1 Theoretical Complexity Analysis.

The total computational complexity of MGMT is governed by three main components: (1) graph-specific Graph Transformer encoders, (2) meta-graph construction, and (3) the final meta-graph Transformer.

Graph-specific Transformer encoders For a graph $\mathcal{G}_i$ with N_i nodes and d-dimensional features, a TransformerConv layer with dense attention costs $\mathcal{O}(N_i^2 d)$. Across n graphs, the total is $\sum_{i=1}^{n} \mathcal{O}(N_i^2 d)$, or $\mathcal{O}(nN^2 d)$ for similar sizes. Standard sparse/linear attention variants can reduce this if needed.

Meta-graph construction Two steps: (a) super-node extraction by scoring and thresholding nodes is $\mathcal{O}(N_i)$ per graph, totaling $\mathcal{O}(nN)$; (b) super-edge creation computes pairwise similarities among selected super-nodes. Let S_i be super-nodes in graph i and $S_{\text{total}} = \sum_i S_i$. This step costs $\mathcal{O}(S_{\text{total}}^2 d)$, i.e., $\mathcal{O}(n^2 S^2 d)$ for roughly S per graph, with $S_i \ll N_i$.

Meta-graph Transformer Applied over S_{total} super-nodes, yielding $\mathcal{O}(S_{\text{total}}^2 d)$ (approximately $\mathcal{O}(n^2 S^2 d)$).

The dominant term is the per-graph encoder, $\sum_i \mathcal{O}(N_i^2 d)$. Meta-graph construction and inference operate on a much smaller set of super-nodes ($S_{\text{total}} \ll \sum_i N_i$) and thus are comparatively lightweight. Quadratic factors at the meta-graph level are in S_{total} (and n), which remains moderate by design.

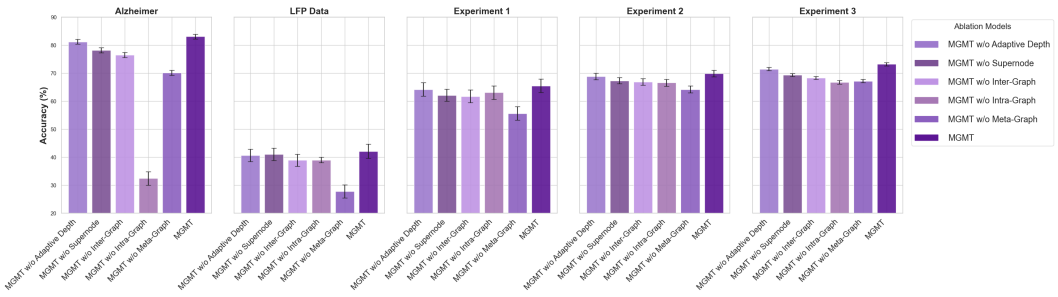

Figure A5: Ablation study results across five datasets evaluating the contribution of each architectural component in the MGMT framework. Each bar shows mean accuracy with standard error (computed over 50 or 100 repetitions depending on the dataset). Removing the adaptive depth selection, supernode selection, or inter-graph edge modeling consistently reduces performance across datasets, underscoring their importance for hierarchical representation learning and cross-graph interaction. Notably, removing intra-graph edges while retaining inter-graph structure leads to a sharp performance drop on the Alzheimer dataset, highlighting the necessity of preserving local structural information. MGMT consistently achieves the highest accuracy, confirming the complementary contribution of all its components.

A9.2 Scalability Analysis

To validate the theoretical complexity discussed in Section A9.1, we empirically evaluated the runtime behavior of MGMT with respect to four key input parameters: number of nodes per graph (N), number of graphs per sample (instance) (n), number of samples, and node feature dimensionality (d). In each experiment, we fixed the model architecture, training epochs (100), and batch size to enable consistent runtime comparisons, and reported runtimes averaged over 10 independent runs. Results in Figure A6 align with theory and show efficient scaling.

Runtime vs. Nodes per Graph (N). As predicted by the $\mathcal{O}(N^2 \cdot d)$ complexity of Transformer-based attention, the observed runtime increases superlinearly with N. The curve aligns closely with a quadratic fit ($R^2 = 0.999$), reflecting the cost of dense all-pairs attention in graph-specific encoders.

Runtime vs. Number of graphs per sample (instance) (n). The runtime grows approximately linearly with n, validating the modular structure of MGMT where graph-specific encoders operate in parallel and the size of the meta-graph remains bounded. This confirms that MGMT scales well with respect to the number of graphs in practical regimes and supports our theoretical analysis in Section A9.1.

Runtime vs. Number of Samples. We observe a near-quadratic growth in runtime (on a log scale) as the number of samples increases, consistent with expectations. This is attributed to repeated forward passes and meta-graph construction across samples, particularly in mini-batch training settings.

Runtime vs. Feature Dimensionality (d). Despite the theoretical linear dependence on d in attention layers, the empirical curve remains nearly flat. This is due to early feature compression in MGMT's architecture, which transforms high-dimensional node features into a lower-dimensional latent space prior to attention and reasoning steps.

A9.3 Runtime Profiling and Model Efficiency

Building on the complexity analysis and scalability trends in Section A9.2, we profile per-epoch runtime to isolate the cost of each architectural component. Table A4 reports average epoch times for MGMT and graph-attention baselines (those that perform graph reasoning and/or meta-graph fusion).

Baselines MGMT's meta-graph reasoning adds minimal overhead: it is faster than MMGL on all datasets except LFP, despite including supernode detection and adaptive depth. Ablations that

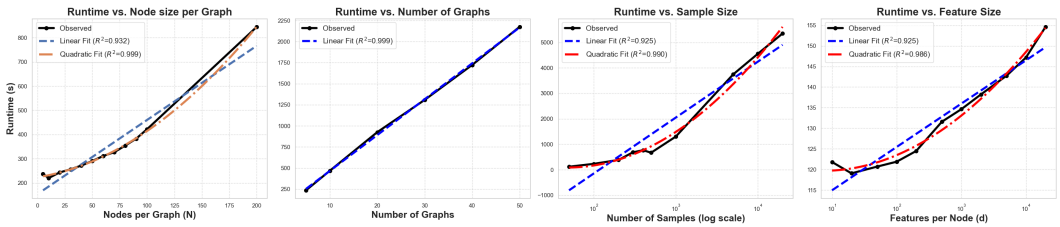

Figure A6: Scalability analysis of MGMT with respect to key input parameters. We evaluate the empirical runtime of MGMT under controlled variations of (a) number of nodes per graph (N), (b) number of graphs per sample (n), (c) number of samples (log scale), and (d) feature dimensionality (d).Runtime scales quadratically with N due to the dense self-attention in the graph-specific Graph Transformers ($\mathcal{O}(N^2 \cdot d)$), and linearly with n, confirming the modular and scalable design of MGMT. Sample size and feature dimension contribute to runtime growth in accordance with expectations, with minor deviations at small scales. Linear and quadratic regression fits are shown for interpretability, along with corresponding R^2 values.

remove intra-graph edges or the meta-graph yield small speedups but reduce accuracy (see Table A2), illustrating a speed–accuracy trade-off.

MultiMoDN, MedFuse, and FlexCare are omitted from Table A4 because they do not use graph representations or attention; direct runtime comparison to graph-based models would be misleading. These methods operate on tabular inputs with shallow fusion, yielding lower computational cost by design but consistently lower accuracy than MGMT (Table A2).

Table A5 decomposes MGMT's epoch time into data preparation, graph encoders, supernode/superedge construction, meta-graph formation, and the final classifier. The dominant cost is the graph Transformer encoder, consistent with the $\mathcal{O}(N^2 d)$ complexity; meta-graph construction and reasoning are comparatively lightweight due to the compact meta-graph.

Overall, MGMT balances expressivity and efficiency: it achieves higher accuracy than non-graph and shallow fusion baselines while maintaining practical per-epoch runtimes.

Compute Infrastructure and Training Cost. All experiments were conducted on a shared CPU-based server provided by our lab. Each training job utilized 4 parallel CPU workers and approximately 4 GB of RAM. No GPU resources were used.

For baseline experiments, we trained a total of 250 models. Each model took on average 5.5 hours to train, amounting to approximately **1,375 CPU hours**.

For MGMT model training and hyperparameter tuning, the total compute time was as follows:

- **LFP dataset:** 100 Optuna trials, each taking 71 minutes on average, resulting in approximately **118.3 CPU hours**
- **Alzheimer dataset:** 100 Optuna trials, each taking 5 hours and 18 minutes on average, resulting in approximately **530 CPU hours**
- **Simulation Setting 1:** 50 iterations, each taking 29 minutes on average, resulting in approximately **24.2 CPU hours**
- **Simulation Setting 2:** 50 iterations, each taking 31 minutes on average, resulting in approximately **25.8 CPU hours**
- **Simulation Setting 3:** 50 iterations, each taking 49 minutes on average, resulting in approximately **40.8 CPU hours**

In total, MGMT-related training required approximately **739 CPU hours**. Additional compute time spent on development, debugging, and model refinement was not recorded.

A10 Sensitivity Analysis of Hyperparameters

The MGMT framework includes several hyperparameters that influence model performance and computational efficiency. In this section, we investigate the sensitivity of two key hyperparameters:

Table A4: Comparison of average epoch runtime (in seconds) between various meta-graph configurations and baseline models across each dataset.

Model Variant	Alzheimer	LFP Data	Experiment 1	Experiment 2	Experiment 3
MMGL	174.23	63.12	21.85	29.0	33.98
MGMT w/o Meta-Graph and Adaptive Depth	174.10	64.33	15.10	17.20	32.60
MGMT w/o Intra-graph Edges	156.77	63.69	15.72	18.83	32.71
MGMT w/o Supernode Selection	215.46	59.61	19.91	19.31	35.61
MGMT	162.93	67.33	16.67	17.59	33.01

Table A5: Detailed epoch running time (in seconds) for the MGMT model across different datasets.

Dataset	Total	Data Prep	Graph-specific encoding	Super-Edge & Node Extraction	Meta-Graph	Final Model
Alzheimer	162.93	1.81	119.24	28.64	1.56	13.18
LFP Data	64.06	0.88	59.74	1.38	1.19	1.25
Experiment 1	16.67	0.23	16.26	0.07	0.06	0.05
Experiment 2	17.59	0.44	16.40	0.26	0.25	0.24
Experiment 3	33.01	0.51	32.25	0.09	0.08	0.08

the attention score threshold (τ) used for supernode selection, and the cosine similarity threshold (γ) used in inter-graph edge construction.

A10.1 Attention Score Threshold (Supernode Selection)

To assess the impact of τ , we conducted a controlled experiment on synthetic data generated under Setting 1 (see Appendix A7). We have a total of 100 samples and 5 graphs per each sample where each graph consisted of 10 nodes, with 30 features per node. We trained all models for 100 epochs and averaged accuracy and runtime over 10 repetitions.

Intuitively, decreasing τ results in more nodes being selected as supernodes, increasing computational cost and potentially introducing noisy or redundant information. In contrast, higher thresholds select fewer supernodes, reducing runtime but possibly discarding useful information. As shown in Figure A7, the runtime decreases steadily as τ increases, which aligns with the reduced number of supernodes and associated computations. However, model accuracy shows a non-monotonic trend: it peaks at $\tau = 0.3$ (64.5%) and declines on either side. This behavior illustrates a tradeoff between overfitting (when too many nodes are included) and information loss (when too few nodes are retained).

A10.2 Cosine Similarity Threshold (Inter-graph Edge Construction)

Moreover, to assess the effect of the cosine similarity threshold γ used for inter-graph edge construction, we performed a controlled sensitivity analysis using synthetic data generated under Setting 1 (see Appendix A7). We have a total of 100 samples and 5 graphs per each sample where each graph consisted of 100 nodes, with 30 features per node. All models were trained for 100 epochs, and both accuracy and runtime were averaged over 10 repetitions.

As shown in Figure A7, runtime remains largely stable across different γ values, indicating that inter-graph edge density has minimal impact on computational overhead since meta-graph construction occurs post graph-specific encoding and operates over a reduced number of supernodes.

Accuracy, however, demonstrates a non-monotonic trend. When γ is very small, the meta-graph becomes fully connected, enabling the model to consider all potential inter-graph interactions. Although this theoretically maximizes expressiveness (since attention-based transformers can learn to prioritize relevant connections), it increases the risk of overfitting due to the inclusion of noisy or spurious edges. On the other hand, when γ is close to 1, the meta-graph becomes sparse or even disconnected, leading to an underutilization of cross-graph dependencies.

The highest accuracy occurs at intermediate values (e.g., $\gamma = 0.4$), suggesting that retaining only the most semantically meaningful inter-graph links allows the model to balance expressiveness with robustness. These findings reinforce the results from our ablation studies (Figure **??**), which demonstrate that incorporating carefully selected inter-graph edges substantially improves downstream performance.

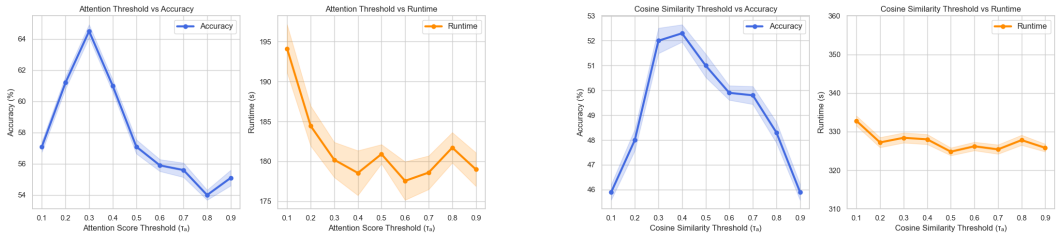

Figure A7: Sensitivity analysis of two key hyperparameters in the MGMT framework. (Left two plots) The attention score threshold τ controls supernode selection. Lower thresholds include more nodes, increasing runtime and potentially introducing noise, while higher thresholds risk discarding informative nodes. Accuracy peaks at $\tau = 0.3$, suggesting a balance between expressiveness and overfitting. (Right two plots) The cosine similarity threshold γ governs inter-graph edge construction in the meta-graph. Accuracy peaks at moderate values of γ, reflecting a trade-off between dense connectivity (risking overfitting) and sparsity (losing cross-graph interactions). Runtime remains largely stable across γ, as meta-graph construction operates over a small number of supernodes.

A11 Impact of Similarity Metrics in Meta-Graph Construction

The construction of inter-graph edges in the meta-graph relies on computing pairwise similarities between node embeddings extracted from different graphs. While cosine similarity is commonly adopted due to its scale-invariant properties, other alternativessuch as Pearson correlation, Euclidean distance, and dot product, may also be used to define similarity across nodes. This section evaluates the extent to which the choice of similarity metric affects downstream performance.

To investigate this, we conducted a controlled experiment on a synthetic dataset generated under Setting 1 (see Appendix A7). For each similarity function, we compute full cross-graph similarity matrices between node embeddings and apply a fixed top-k rule with $k = 10$ to select inter-graph edges, ensuring identical sparsity across metrics. Each configuration is run 50 times; we report mean accuracy.

We compare cosine similarity, Pearson correlation, negative Euclidean distance converted to similarity via $1/(1 + d_{ij})$, and dot product. Results show modest but consistent differences: dot product attains the highest accuracy (0.661), followed by Pearson (0.654), Euclidean (0.648), and cosine (0.642). The spread is small (1.9 percentage points), indicating limited sensitivity to the similarity choice under this setup.