Knowledge Graph-Based Multi-Agent System for Fertilizer Optimization

Nahed Abu Zaid, Jaya Shruti Chintalapati, Mihir Shah, Ranga R. Vatsavai
Department of Computer Science, North Carolina State University, Raleigh, NC, USA
{naabuzai, jchinta, mshah25, rrvatsav}@ncsu.edu

***Abstract*—Precisely measuring the use of resources in farming for sustainability improvement is the horizon of modern technologies in precision agriculture. Traditional methods of fertilizer application depend on static models, which cannot account for environmental changes in real time. Hence, they result in inefficient nutrient use and adverse ecological impacts. We propose a Knowledge Graph (KG)-based Multi-Agent System (MAS) that integrates Graph Neural Networks (GNNs), and real-time reasoning to dynamically optimize fertilizer recommendations based on soil, weather, and crop conditions. Our system, KG-MASFO, constructs a unified KG from heterogeneous agricultural data, applies a domain-specialized multi-agent GNN architecture, and uses a lightweight fusion module to synthesize agent outputs. KG-MASFO supports edge deployment and includes a rule-based, post-hoc explanation component that interprets predictions using contextual semantic cues and produces human-readable rationales. Compared to our implemented models we see that KG-MASFO achieves superior performance with a Mean Squared Error of 0.0020 and an R-squared score of 0.9739, confirming its high accuracy and generalizability. This work offers a scalable, interpretable, and adaptive solution to fertilizer optimization, minimizing environmental impact while maximizing crop yield. Future enhancements will focus on advanced agent coordination and temporal modeling for long-term decision support.**

***Index Terms*—Predictive AI, Graph Neural Networks, Multi-Agent Systems, Game Theory, Edge Computing,Explainable AI, Real-Time Reasoning, Precision Agriculture**

I. Introduction

Agriculture plays a fundamental role in sustaining human life, yet optimizing fertilizer application remains a persistent challenge [1], [2]. Farmers must balance crop nutrient requirements, environmental sustainability, and economic feasibility to ensure efficient and responsible farming practices [3], [4]. Traditional fertilizer application methods often rely on static, rule-based models that provide generalized recommendations without accounting for dynamic environmental conditions, leading to inefficient nutrient utilization. Excessive fertilizer use contributes to nutrient runoff and environmental degradation, such as the formation of hypoxic zones in downstream water bodies [5], [6]. Conversely, insufficient fertilizer application results in nutrient deficiencies that can negatively impact crop growth and yield [7]. Addressing these limitations requires a dynamic, data-driven system capable of adapting fertilizer recommendations in real time, based on integrated observations of soil composition, weather variability, and crop growth stages [8], [9]. ABC

Lets consider our motivating example as shown in Fig. 1. The left side of the figure depicts a farmer using a conventional approach in which solid fertilizer pellets (represented as orange balls) are distributed uniformly across a field without accounting for soil variability, weather fluctuations, or crop-specific nutrient needs. This indiscriminate application method is widely adopted but highly inefficient. In the scenario where rainfall follows fertilizer application, excess nutrients on the soil surface are washed into nearby water bodies, leading to nutrient runoff that contributes to water pollution, eutrophication, and the degradation of aquatic ecosystems. In contrast, in dry conditions, the fertilizer remains underutilized due to insufficient soil moisture, resulting in stunted crop growth and inefficient nutrient absorption. These inefficiencies increase input costs for farmers while simultaneously posing environmental risks, reinforcing the need for a more adaptive and efficient fertilizer application strategy. The right side of Fig. 1 presents an alternative approach that leverages a network of AI-driven agents to optimize fertilizer use. By integrating data from real-time soil sensors, weather reports, and agronomic models, the system enables precise, data-driven fertilizer decisions. The multi-agent system, consisting of specialized agents such as the Weathering Agent, Crop Agent, and Soil Agent, collaborates to analyze dynamic agricultural conditions and adjust fertilizer application accordingly. Unlike traditional systems, which rely on uniform fertilizer distribution, this approach ensures that fertilizers are applied only where and when they are most needed. As a result, the system minimizes nutrient runoff, optimizes soil nutrient retention, and enhances crop growth, ultimately improving both economic and environmental sustainability.

Improper fertilizer application is but one factor, leading to tremendous ecological ramifications on regional and national scales. In the Midwest of the United States, home to large-scale mechanized farming, conventional fertilization often ignores the spatial soil heterogeneity and temporal variability of weather. [6]. Research conducted by Kellogg Biological Station Long Term Ecological Research (LTER) program shows that nitrogen fertilizer remarkably alters soil nitrogen dynamics and therefore impairs other crucial ecosystem services[10].

One critical outcome of this mismanagement is that during periods of heavy rainfall, surplus fertilizer is rapidly transported from agricultural fields into surrounding waterways. These nutrients ultimately flow into the Gulf of Mexico, where they promote the development of expansive hypoxic “dead zones”—low-oxygen aquatic regions that can grow to sizes comparable to the state of New Jersey [11], [5]. The Midwest

is particularly impacted, as excessive fertilizer use contributes significantly to the release of nitrous oxide (N_2O)—a greenhouse gas nearly 300 times more potent in radiative efficiency than carbon dioxide (CO_2)—and accounts for over 50% of total N_2O emissions from U.S. agriculture, making fertilizer overuse a key driver of climate change [12].

As mitigation measures for the adverse environmental effects of excessive fertilizer application, the suggested programs include compensation and incentives for farmers to adopt precision fertilization methods. [13]. Although it was hoped that these programs would promote sustainable agricultural practices, their effectiveness has been hindered by an unwillingness in the agricultural sector to accept regulatory supervision combined with continuous technological and economic barriers to its wide adoption[10]. As a result, there is an urgent need for novel approaches that go beyond static prescriptions toward the dynamic setting of fertilizer use through intelligent adaptive systems. Such a system must be able to ingest real-time environmental data and provide automated, data-driven recommendations for agricultural efficiency and environmental sustainability.

Many precision agriculture technologies came into practice addressing the traditional inefficiency of fertilization through the use of an AI-ended fertilization system. For example, John Deere's ExactShot, which essentially applies sensor-based robotics to putting fertilizer on individual seeds, achieves an excess of 60% [14]. While they mark big milestones in resource efficiency from application to application, those technologies fail to keep pace with real-time environmental dynamicization. Or, multi-agent optimization frameworks can't coordinate such decisions via those approaches. In a parallel line to these innovations, machine learning-based approaches have also been applied to soil-quality assessment and fertilization recommendations [4]. Agricultural models enhanced with Knowledge Graphs (KGs) also hold promise in representing complex relationships among soil, crop, and climate variables [7], [15].

This is one of the biggest issues with those methods: they are static or have a very narrow scope, making them generally unable to respond to needs for real-time and adaptive decision-making under uncertainty [15], [16]. Progress has been made in AI-assisted agriculture, but it is the limitations among these approaches that dominate their discussion. Most models base their predictions purely on historical data with no awareness of changing environmental context [4], and thus they are ineffective in fast-changing environments. Most approaches treat fertilizer application as a statically optimized process based on fixed conditions, ignoring dynamic processes such as soil composition shifts, weather fluctuations, and crop growth stages [17]. A second major limitation is the lack of collaborative multi-agent frameworks capable of integrating decisions across all these variables [16]. Finally, the lack of explainability (XAI) continues to hinder farmer adoption, as users often require transparent justifications before trusting AI-driven decisions [15].

The above-mentioned issues need to be addressed by a rather advanced adaptive and interpretable AI system that puts all the possible environmental data and optimizes fertilizer

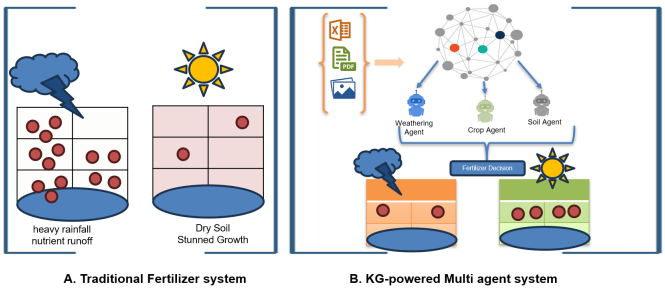

Fig. 1. Comparison between a traditional fertilizer application approach (left), which applies nutrients uniformly regardless of environmental variability, and an intelligent, data-driven method (right) that leverages real-time inputs from soil, crop, and weather domains to make adaptive, context-aware fertilizer decisions.

recommendations using a multi-agent framework. This paper presents KG-MASFO, an intelligent AI-powered multi-agent system for fertilizer optimization. It uses GNNs to determine what are the nutrient needs of soils, weather conditions, and crop requirements, in addition to a multi-agent reasoning framework, which controls the decision logic based on things like soil, crop, and weather. It integrates real-time sensor data and uses edge-computing so that efficient infield fertilizer optimization can be done. The system uses an AI explainability system to provide transparency and interpretability to these decisions, so farmers can understand, and ultimately, adopt these technologies when they use them in practice. KG-MASFO-that is, AI, real time data integration and multi-agent decision making-ultimately presents a scalable and adaptable approach to precision fertilization. This research aims to empower agricultural producers while simultaneously addressing environmental sustainability challenges such as making nutrient runoff much lower and reducing greenhouse gas emissions. KG-MASFO is a leap towards precision agriculture by moving unknown by moving farmers from traditional static fertilization practices to intelligent data-driven practices.

Our specific contributions in this paper are as follows:

1) We propose *KG-MASFO*, a domain-agnostic AI system that integrates Graph Neural Networks (GNNs) with a modular multi-agent architecture to enable real-time fertilizer optimization based on heterogeneous agricultural data.
2) We design a scalable *multi-agent prediction pipeline*, where specialized GNN-based agents (e.g., soil, crop, and weather) operate on semantically typed subgraphs and collaboratively contribute to an interpretable fusion model. The architecture supports edge deployment and sensor-driven updates.
3) We develop a lightweight, symbolic *post-hoc explanation module* that interprets input features and system predictions using rule-based mappings and structured templates.

The remainder of the paper is structured as follows: Section II reviews related work Section III defines key terminology and formalizes the problem statement. Section IV introduces the proposed KG-MASFO approach. Section V details the

experimental setup and evaluation, with Section V-E presenting the experimental results and their analysis. Finally, Section VI concludes the paper and suggests directions for future work.

II. Related Work

This work will actively contribute to three major research lines in smart agriculture: Precision Farming application of artificial intelligence (AI), agricultural KGs for structured agricultural intelligence, and Multi-Agent Systems (MAS) for farm management decision-making in a distributed way.

AI has been widely applied in agriculture today for increased productivity, sustainability, and resource utilization. Applications include crop monitoring, resource distribution, decision support systems, and fully autonomous agricultural operations. Notably, machine learning models have shown tremendous success in the areas of soil analysis and crop yield prediction [15]. However, current AI-driven agricultural systems often depend on static historical datasets and lack the ability to adapt in real time to evolving environmental conditions—limiting their suitability for dynamic fertilizer optimization [15].

KGs have emerged as powerful tools for integrating heterogeneous agricultural data, therefore providing an understanding of complex agri-environmental systems in a more structured and interconnected manner. KGs effectively describe the relationship between soil types, weather parameters, crop classes, and fertilizers to aid intelligent decision-making[7]. A noteworthy example is CropDP-KG, which, by extending natural language processing (NLP) techniques, merges the data on crop pests and diseases and hence shows how KGs can provide real value to agriculture [7].

The multiagent systems (MAS) have been alternatively looked into as a means for decentralizing agricultural decision-making and automating it. MAS architectures can incorporate data from field sensors, drones, and satellites to deliver hyper-localized and task-specific recommendations[18]. Recent work has extended MAS to optimize the distribution of resources and scheduling of machinery among farms. A drawback of these advancements is that many current MAS implementations are task-specific and unable to long-term predictive modeling of potential multiple objectives involved in holistic farm management[16].

Recent advancements in knowledge graph based learning and attention driven neural models further validate the potential of our approach. Notably, the KGANSynergy framework [19] demonstrated the effectiveness of integrating hierarchical propagation and multi-head attention over biomedical KGs for drug synergy prediction. By encoding multihop neighborhood relations between drug entities, proteins, and cell lines, KGANSynergy successfully captured rich, multisource biological knowledge and improved interpretability through attention weights. This supports the argument that high order neighborhood semantics, when aggregated through structured KGs, significantly enhance predictive capabilities in complex, data dense domains like bioinformatics and precision medicine. We adapt these ideas into the agricultural domain by modeling soil-crop-weather interactions using a structured agri-KG and agent specific attention networks.

In parallel, MARLPaR [Li et al., 2018] has emphasized the role of multi-agent reinforcement learning for knowledge graph path reasoning. Their system jointly trains a relation selection agent and an entity selection agent to navigate knowledge graphs for query answering tasks. This dual-agent strategy significantly improves inference quality, especially in cases with 1-N/N-N relations. While their application domain lies in general KGs and question answering, our system applies similar multi-agent principles in the agriculture domain. In particular, our agents (SoilAgent, WeatherAgent, and CropAgent) independently reason over domain-specific subgraphs before their outputs are fused via the LAMMA reasoning layer. Although we do not use reinforcement learning in our current approach, the design philosophy of MARLPaR strongly supports our agent-based decomposition for interpretable, domain specialized KG reasoning.

Our work helps overcome the limitations in the above proposal by suggesting an integrated framework that brings together interpretable artificial intelligence, dynamically changing KGs, and a whole MAS architecture. The synergistic approach is transparent in decision-making, provides a wide scope of adaptability to real-time agricultural dynamics, and builds a scalable yet sustainable solution for intelligent fertilizer optimization in precision agriculture.

III. Problem Statement

In this work, we address the challenge of optimizing fertilizer application in precision agriculture by integrating heterogeneous environmental data into a structured, intelligent system. We define a *precision agriculture knowledge graph* as $\mathcal{G} = (\mathcal{V}, \mathcal{T}, \mathcal{E}, \mathcal{L}, \phi)$, where $\mathcal{V}$ is the set of entities (e.g., soil samples, crop types, weather records), $\mathcal{T}$ denotes entity types, and $\mathcal{E}$ is the set of typed relationships between entities. The labeled triples are expressed as $\mathcal{L} \subseteq \mathcal{V} \times \mathcal{E} \times \mathcal{V}$, and the function $\phi : \mathcal{E} \to \mathcal{T}$ assigns semantic types to relationships.

A graph instance p represents a multi-hop semantic path through the knowledge graph:

$$p = (v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \cdots \xrightarrow{r_k} v_k) \tag{1}$$

where $v_i \in \mathcal{V}$ are domain-specific entities and $r_i \in \mathcal{E}$ capture their semantic connections. Such paths reflect agricultural interdependencies relevant to fertilizer decisions.

Given structured sources $\mathcal{D}_s$, $\mathcal{D}_c$, and $\mathcal{D}_w$ representing soil, crop, and weather domains respectively, we construct a unified knowledge graph $\mathcal{G}$. On top of this graph, we develop a modular pipeline $\mathcal{M} = \{\text{GNN}, \text{Single-Agent}, \text{Multi-Agent}, \text{Reasoning}\}$ that evolves from domain-specialized GAT-based learning to coordinated multi-agent prediction, culminating in symbolic post-hoc explanations derived from input features and agronomic rules. Formally, our goal is to (1) build a semantically rich knowledge graph $\mathcal{G}$ from heterogeneous agricultural datasets $\{\mathcal{D}_i\}$, (2) train specialized agents to predict optimal fertilizer values using *localized graph features*—i.e., subgraph-level patterns relevant to soil, crop, and weather—and (3) generate interpretable, scenario-specific justifications to support decision-making.

Three criteria need to be satisfied by our approach: (i) schema-aligned graph construction preserves the semantics of source data, (ii) attention-based GNN learning minimizes cross-domain dependencies, and (iii) reasoning must be clear and human-oriented and must rely on agronomic knowledge.

IV. Methodology

We now introduce KG-MASFO, an end-to-end, modular pipeline to facilitate context-aware fertilizer recommendations based on a Knowledge Graph (KG), predictive modeling based on Graph Neural Networks (GNNs), and an MAS-based coordination layer. Our architecture follows a three-phase methodology: (1) KG construction, (2) GNN learning, (3) and MAS-based decision support and refinement.

A. Phase 1: Knowledge Graph Construction from Heterogeneous Agricultural Data

1) Data Source Characterization and Preprocessing: Our pipeline begins with four distinct data sources: *Soil*, *Weather*, *Crop*, and *Fertilizer*. The datasets are gathered from different sources with different type of format, such as text, image, or spreadsheet data. It is transformed separately into a tabular table format, where rows show instances (e.g., soil samples, crop observations) and columns show their respective attributes.

a) Preprocessing Workflow: To ensure consistency and semantic alignment across heterogeneous sources, we apply the following preprocessing steps:

- Manual cleaning: Noisy entries, such as placeholder strings (e.g., "N/A", or "—", are identified and removed manually during data collection.
- Categorical Encoding: For some textual or categorical fields (e.g., soil texture, crop variety), we used a lookup table to map those terms into unified ones to avoid syntax or semantic misalignment across datasets, especially when synonyms are used, for example we mapped the term "silty loam" to "loam" type.
- Imputation of Missing Data: To handle missing values (like precipitation and temperature), we used simple mean imputation methods. The missing numeric entries were replaced with the column-wise mean of all available data points.
- Outlier Filtering: We removed such impossible physically related data such as negative concentrations of nutrients, below 1 pH, or above 100°C temperatures.

b) Schema Mapping.: Following the preprocessing stage, Each dataset is aligned to a unified schema using a configurable dictionary, which assigns semantic roles to columns (e.g., node or edge types) and thus allows the system to generalize over multiple domains. For example: The *crop_type* field can possibly be related to a node type named *Crop*. The field *soil_texture* may define an edge *type: hasTexture*. The dictionary further supports the three-part frameworks given as (SubjectType, Predicate, ObjectType), which are then used in edge formation to infer semantic relationships between datasets.

2) KG Construction: From Tabular Data to Typed Nodes and Edges: We now describe the KG construction process, shown in Algorithm 2, which takes as input a set of structured tables $\mathcal{T} = \{T_1, T_2, ..., T_n\}$, each representing a distinct entity type (e.g., soil, crop, fertilizer). The goal is to produce a unified typed knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ contains typed nodes and $\mathcal{E}$ contains semantically labeled edges between them.

The algorithm begins by initializing empty sets for nodes $\mathcal{V}$, edges $\mathcal{E}$, and a registry map $\mathcal{R}$ to avoid node duplication (line 4). It then iterates over each input table $T_i \in \mathcal{T}$, invoking `Type` to infer the entity type τ_i, and `Key` to infer the primary key column k_i (line 6). It then loops over each row $r \in T_i$ to retrieve the unique key value $v = r[k_i]$ (line 8) and query if it is present in the registry (line 9). If not, it creates a node from the whole row and gives it type τ_i with $\texttt{Node}(\tau_i, r)$ (line 10). The node is then added to the node set $\mathcal{V}$, and the registry is updated accordingly (line 11). Since the registry $\mathcal{R}$ is implemented as a hash map, this lookup operation takes constant time $O(1)$, in contrast to a linear scan over $\mathcal{V}$, which would require $O(n)$.

In the second phase (lines 13–17) the algorithm constructs semantically meaningful triples by connecting typed entities across tables. For every pair of tables (T_i, T_j), it iterates over row pairs (r_i, r_j), and checks if a logical connection can be made. We use this linkage rule:

$$\texttt{Link}(r_i, r_j) = \begin{cases} \text{true}, & \text{if } \exists\, (\tau_i, \rho, \tau_j) \in \Sigma \text{ and } r_i[a] = r_j[b] \\ \text{false}, & \text{otherwise} \end{cases}$$

where we define Σ a schema configuration of allowed semantic mappings (τ_i, ρ, τ_j), and the match condition $r_i[a] = r_j[b]$ ensures entity correspondence (e.g., shared crop ID or location).

If the condition holds (line 15) the algorithm infers the relationship type $\rho = \texttt{RelType}(r_i, r_j)$ from the same schema (line 16), and constructs a typed triple (line 17). Each such triple captures a meaningful fact — for instance: `(Fertilizer_A, applied_to, Soil_B)`. Once all edges are constructed, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is persisted via `Store` (line 18) using the Neo4j graph database [20], and returned (line 19).

B. Phase 2: GNN Learning: Fertilizer Prediction via Graph Attention

We now describe the graph-based learning phase, shown in Algorithm 29, which takes as input a typed knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and learns a node-level regression model for predicting a target attribute t (e.g., fertilizer amount) using Graph Attention Networks (GATv2).

The process begins by converting the structured graph into numerical tensors. From each node $v_i \in \mathcal{V}$, we extract all numeric attributes into a matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where d is the number of features per node (line 4). The target attribute t is isolated into a label vector $\mathbf{y} \in \mathbb{R}^{|\mathcal{V}| \times 1}$ (line 5). To ensure consistent feature scaling, both $\mathbf{X}$ and $\mathbf{y}$ are normalized using MinMax normalization (line 6). The set of typed relations $\mathcal{E}$ is transformed into a sparse index tensor $\mathbf{E} \in \mathbb{N}^{2 \times |\mathcal{E}|}$, which preserves the graph's edge structure (line 7). These components are then encapsulated into a `torch_geometric.data.Data`

Algorithm 1: KG Construction

Input: Structured tables $\mathcal{T} = \{T_1, \ldots, T_n\}$
Output: Unified typed knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Step A: Node Construction
$\mathcal{V}, \mathcal{E}, \mathcal{R} \leftarrow \emptyset$
foreach $T_i \in \mathcal{T}$ **do**
  $\tau_i \leftarrow \texttt{Type}(T_i),\ k_i \leftarrow \texttt{Key}(T_i)$
  foreach $r \in T_i$ **do**
    $v \leftarrow r[k_i]$
    if $v \notin \mathcal{R}$ **then**
      $v \leftarrow \texttt{Node}(\tau_i, r)$
      $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\},\ \mathcal{R}[v] \leftarrow v$
Step B: Edge (Triple) Construction
foreach $(T_i, T_j) \in \mathcal{T} \times \mathcal{T}$ **do**
  foreach $(r_i, r_j) \in T_i \times T_j$ **do**
    if $\texttt{Link}(r_i, r_j)$ **then**
      $\rho \leftarrow \texttt{RelType}(r_i, r_j)$
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\mathcal{R}[r_i.\text{id}], \rho, \mathcal{R}[r_j.\text{id}])\}$
$\texttt{Store}(\mathcal{V}, \mathcal{E})$
return $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

Fig. 2. Schema-driven graph generation over tabular data. Each table T_i yields nodes of type τ_i; edges are inferred via relational mapping (τ_i, ρ, τ_j).

Algorithm 2: Graph Neural Network Training for Node-Level Regression

Input: KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, target attribute t
Output: Trained GATv2 model $f_\theta : \mathbb{R}^d \to \mathbb{R}$
Step A: Graph Tensor Construction
$\mathbf{X} \leftarrow \texttt{ExtractFeatures}(\mathcal{V}) \in \mathbb{R}^{|\mathcal{V}| \times d}$
$\mathbf{y} \leftarrow \texttt{ExtractTarget}(\mathcal{V}, t) \in \mathbb{R}^{|\mathcal{V}| \times 1}$
$\mathbf{X} \leftarrow \texttt{MinMaxNormalize}(\mathbf{X})$,
  $\mathbf{y} \leftarrow \texttt{MinMaxNormalize}(\mathbf{y})$
$\mathbf{E} \leftarrow \texttt{BuildEdgeIndex}(\mathcal{E}) \in \mathbb{N}^{2 \times |\mathcal{E}|}$
$\mathcal{D} \leftarrow \texttt{GraphData}(\mathbf{X}, \mathbf{E}, \mathbf{y})$
Step B: GNN Model Definition
Define multi-head GATv2: $f_\theta = \mathcal{F}_L^{\text{GAT}}(\mathbf{X}, \mathbf{E})$
$\mathcal{L} \leftarrow \texttt{SmoothL1Loss}(), \mathcal{O} \leftarrow \texttt{AdamW}(f_\theta, \eta)$
Initialize: $\mathcal{L}_{\text{best}} \leftarrow \infty$, patience $\leftarrow P$, counter $\leftarrow 0$
Step C: Training Loop with Early Stopping
for $e = 1$ *to* T **do**
  $f_\theta.\texttt{train}(), \mathcal{O}.\texttt{zero\_grad}()$
  $\hat{\mathbf{y}} \leftarrow f_\theta(\mathbf{X}, \mathbf{E})$
  $\mathcal{L}_{\text{cur}} \leftarrow \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
  $\mathcal{L}_{\text{cur}}.\texttt{backward}(); \mathcal{O}.\texttt{step}()$
  if $\mathcal{L}_{cur} < \mathcal{L}_{best}$ **then**
    $\mathcal{L}_{\text{best}} \leftarrow \mathcal{L}_{\text{cur}}$, counter $\leftarrow 0$
    $\texttt{SaveModel}(f_\theta)$
  else
    counter $\leftarrow$ counter + 1
  if *counter* $\geq P$ **then**
    break
Step D: Inference and Evaluation
Reload best checkpoint θ^*
$\hat{\mathbf{y}} \leftarrow f_{\theta^*}(\mathbf{X}, \mathbf{E})$
return Optimized f_{θ^*}

object: $\mathcal{D} = \texttt{Data}(x = \mathbf{X}, edge\_index = \mathbf{E}, y = \mathbf{y})$ (line 8), which serves as the training input for the GNN. The object $\mathcal{D}$ captures all graph-relevant signals—node features, edge connectivity, and supervision labels—in a format directly consumable by GATv2 layers.

A multi-layer GATv2 model f_θ is then constructed (line 10), consisting of stacked attention heads that perform neighborhood aggregation while weighting the importance of each neighbor dynamically. Each layer transforms node features through learned attention coefficients, capturing structural dependencies and feature salience jointly [21]. The training loss is defined using Smooth L1 Loss [22], a robust loss function that balances sensitivity and outlier resistance, and optimized with AdamW [23] using learning rate η (line 11). To avoid overfitting, early stopping is enforced by tracking the best observed loss $\mathcal{L}_{\text{best}}$ and halting training when no improvement occurs within a window of P epochs (line 12).

During training (lines 14–12), the model predicts outputs $\hat{\mathbf{y}}$ for the entire graph, computes the current loss $\mathcal{L}_{\text{cur}}$, and performs a backward update. If $\mathcal{L}_{\text{cur}}$ is lower than the previously recorded best loss, the model checkpoint is updated and the counter reset. Otherwise, the counter increments, and training stops if it exceeds the patience threshold.

Once training completes the model reloads the checkpoint corresponding to the lowest observed validation loss, denoted by the optimal parameter set θ^* (line 27). This checkpoint represents the best-performing instance of the multi-head GATv2 model, trained over the graph structure and normalized attributes. The model is then applied to the full input graph—comprising the node feature matrix $\mathbf{X}$ and edge index tensor $\mathbf{E}$—to compute the predicted output $\hat{\mathbf{y}} = f_{\theta^*}(\mathbf{X}, \mathbf{E})$. Finally, the trained model f_{θ^*} is returned (line 29) for downstream evaluation or integration within a larger inference system.

C. Phase 3: Agents Based Decision Support and Model Refinement

1) Single-Agent GATv2-Based Training on Typed Subgraphs: We now describe the training process for a single-agent GATv2 model, as detailed in Algorithm 28. This process focuses on learning localized embeddings over a semantically typed subgraph $\mathcal{G}_\tau = (\mathcal{V}_\tau, \mathcal{E}_\tau)$, where τ refers to a specific agent type such as *Soil*, *Crop*, or *Weather*. The goal is to train a dedicated model f_{θ_τ} that learns domain-specific patterns from this subgraph and predicts a continuous target attribute t.

The process begins by extracting numerical features from all nodes $v \in \mathcal{V}_\tau$, resulting in a dense feature matrix $\mathbf{X}_\tau \in \mathbb{R}^{|\mathcal{V}_\tau| \times d}$, where each row corresponds to a node and each column to a normalized input variable (line 4). From this matrix, the target

variable is extracted as a column vector $\mathbf{y}_\tau \in \mathbb{R}^{|\mathcal{V}_\tau|\times 1}$ (line 5). Both $\mathbf{X}_\tau$ and $\mathbf{y}_\tau$ are scaled using MinMax normalization:

$$\texttt{MinMax}(z_i) = \frac{z_i - \min(z)}{\max(z) - \min(z)}$$

This ensures that all features and targets lie in a common numerical range $[0, 1]$, preventing any single variable from dominating the optimization process (line 6).

The edge set $\mathcal{E}_\tau$ is then encoded into a sparse integer tensor $\mathbf{E}_\tau \in \mathbb{N}^{2\times|\mathcal{E}_\tau|}$, where each column represents a directed edge between two nodes (line 7). All three components—features, edges, and labels—are encapsulated into a PyTorch Geometric graph object $\mathcal{D}_\tau = (\mathbf{X}_\tau, \mathbf{E}_\tau, \mathbf{y}_\tau)$, which serves as the input to the GNN (line 8).

The model architecture f_{θ_τ} is initialized as a stack of multi-head GATv2 layers (line 10). Each layer applies an attention mechanism over neighboring nodes. Specifically, each node i computes a learned attention coefficient $\alpha_{ij}^{(k)}$ for every neighbor j, and aggregates messages according to the multi-head attention formula [21], as introduced below:

$$\mathbf{h}_i^{(l+1)} = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j^{(l)}\right)$$

where $\|$ denotes concatenation over K attention heads, σ is a non-linear activation (e.g., ELU), and $\mathbf{W}^{(k)}$ are learnable projection matrices.

We use the Smooth L1 loss function $\mathcal{L}$ = `SmoothL1Loss()` [22], which is more stable than MSE under noisy targets, and optimize using AdamW [23] with learning rate η and weight decay (line 11). A best-loss tracker $\mathcal{L}_{\text{best}}$, a patience counter, and a maximum epoch limit T are also initialized (line 12).

In the training phase (line 14), the model is iteratively updated over epochs. Each step involves a forward pass over the graph to compute predictions $\hat{\mathbf{y}}_\tau = f_{\theta_\tau}(\mathbf{X}_\tau, \mathbf{E}_\tau)$, loss computation, and backpropagation. The checkpointing mechanism ensures that only the best-performing weights θ_τ^* are retained, while the patience counter prevents overfitting by terminating training early if no improvement is observed over P consecutive epochs. At the end of training, the best weights are reloaded (line 26) and used to compute the final predictions. The output is a trained GATv2 model $f_{\theta_\tau^*}$ specialized for the subgraph $\mathcal{G}_\tau$, capturing both semantic structure and numeric dependencies within that domain.

2) Multi-Agent GNN-Based Prediction: To enable structured learning over heterogeneous graphs, we implement multi-agent prediction architecture as defined in Algorithm 25. The core idea is to partition the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into n semantic subgraphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$, each corresponding to a specific view, type, or feature space. These subgraphs are independently processed by specialized GNN-based agents, whose outputs are subsequently fused into a unified prediction.

The pipeline begins by preprocessing each subgraph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ (line 4). For every agent i, we extract its input feature matrix $\mathbf{X}_i \in \mathbb{R}^{|\mathcal{V}_i|\times d}$, target vector $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{V}_i|\times 1}$, and sparse edge index $\mathbf{E}_i \in \mathbb{N}^{2\times|\mathcal{E}_i|}$. All features and labels are normalized via MinMax scaling (lines 6–7), and the resulting tensors are encapsulated into a graph object $\mathcal{D}_i = (\mathbf{X}_i, \mathbf{E}_i, \mathbf{y}_i)$ (line 9) for downstream learning.

Algorithm 3: Domain-Adapted GATv2 Training for Single-Agent Embedding

1 **Input** $\mathcal{G}_\tau = (\mathcal{V}_\tau, \mathcal{E}_\tau)$, target variable t
2 **Output** Trained GATv2 model f_{θ_τ} for agent type τ
3 **Step A: Data Preparation for Specialized Agent** τ
4 Extract numerical features from all $v \in \mathcal{V}_\tau$
5 Construct feature matrix $\mathbf{X}_\tau \in \mathbb{R}^{|\mathcal{V}_\tau|\times d}$, target vector $\mathbf{y}_\tau \in \mathbb{R}^{|\mathcal{V}_\tau|\times 1}$
6 Normalize: $\mathbf{X}_\tau, \mathbf{y}_\tau \leftarrow$ `MinMaxScaler(·)`
7 Build sparse index tensor: $\mathbf{E}_\tau \in \mathbb{N}^{2\times|\mathcal{E}_\tau|}$
8 $\mathcal{D}_\tau \leftarrow$ `GraphData`$(\mathbf{X}_\tau, \mathbf{E}_\tau, \mathbf{y}_\tau)$
9 **Step B: Initialize GATv2 for Agent Type** τ
10 Define GATv2 model $f_{\theta_\tau} = \mathcal{F}^{\text{GAT}}(\mathbf{X}_\tau, \mathbf{E}_\tau)$
11 Set optimizer $\mathcal{O} \leftarrow$ `AdamW`(f_{θ_τ}, η), loss $\mathcal{L} \leftarrow$ `SmoothL1Loss()`
12 Track best loss: $\mathcal{L}_{\text{best}} \leftarrow \infty$, counter $\leftarrow 0$, patience $\leftarrow P$
13 **Step C: Train Agent-Specific GNN with Early Stopping**
14 **for** $e = 1$ *to* T **do**
15 Set f_{θ_τ}.`train()`, $\mathcal{O}$.`zero_grad()`
16 $\hat{\mathbf{y}}_\tau \leftarrow f_{\theta_\tau}(\mathbf{X}_\tau, \mathbf{E}_\tau)$
17 $\mathcal{L}_{\text{cur}} \leftarrow \mathcal{L}(\hat{\mathbf{y}}_\tau, \mathbf{y}_\tau)$
18 $\mathcal{L}_{\text{cur}}$.`backward()`, $\mathcal{O}$.`step()`
19 **if** $\mathcal{L}_{cur} < \mathcal{L}_{best}$ **then**
20 `SaveModel`(f_{θ_τ}), $\mathcal{L}_{\text{best}} \leftarrow \mathcal{L}_{\text{cur}}$, counter $\leftarrow 0$
21 **else**
22 counter $\leftarrow$ counter +1
23 **if** *counter* $\geq P$ **then**
24 `break`
25 **Step D: Inference**
26 Reload best checkpoint θ_τ^*
27 $\hat{\mathbf{y}}_\tau \leftarrow f_{\theta_\tau^*}(\mathbf{X}_\tau, \mathbf{E}_\tau)$
28 **return** Trained agent model $f_{\theta_\tau^*}$

Each agent $f_i = \mathcal{F}_i^{\text{GNN}}(\cdot)$ is defined as a GNN encoder (e.g., GAT, GCN), parameterized independently to operate over its assigned subgraph $\mathcal{G}_i$ (line 12). The outputs of all agents are vector embeddings $\mathbf{Z}_i = f_i(\mathbf{X}_i, \mathbf{E}_i)$, each representing the agent-specific representation space. These embeddings are concatenated into a global representation:

$$\mathbf{Z} = [\mathbf{Z}_1 \| \mathbf{Z}_2 \| \ldots \| \mathbf{Z}_n] \in \mathbb{R}^{|\mathcal{V}|\times(n\cdot d')}$$

as shown in line 19. The fused vector $\mathbf{Z}$ is then fed into a shallow neural network f_{fuse}, which produces the final prediction $\hat{\mathbf{y}} = f_{\text{fuse}}(\mathbf{Z})$ (line 20).

The model is optimized end-to-end using the Smooth L1 loss $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ and AdamW optimizer (line 14). We adopt early

Algorithm 4: Domain-Agnostic Multi-Agent GNN Prediction

1 **Input** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, target variable y
2 **Output** Trained fusion model f_{fuse} with agent set $\{f_1, \ldots, f_n\}$
3 **Step A: Preprocessing for Each Agent Domain**
4 Partition $\mathcal{G}$ into n disjoint or overlapping typed subgraphs $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$
5 **foreach** *agent* $i = 1, \ldots, n$ **do**
6 Extract numeric features $\mathbf{X}_i \in \mathbb{R}^{|\mathcal{V}_i| \times d}$ and labels $\mathbf{y}_i \in \mathbb{R}^{|\mathcal{V}_i| \times 1}$
7 Normalize: $\mathbf{X}_i, \mathbf{y}_i \leftarrow \texttt{MinMaxScaler}(\cdot)$
8 Construct edge tensor $\mathbf{E}_i \in \mathbb{N}^{2 \times |\mathcal{E}_i|}$
9 $\mathcal{D}_i \leftarrow \texttt{GraphData}(\mathbf{X}_i, \mathbf{E}_i, \mathbf{y}_i)$
10 **Step B: Agent and Fusion Initialization**
11 **foreach** *agent* $i = 1, \ldots, n$ **do**
12 Define $f_i = \mathcal{F}_i^{\text{GNN}}(\mathbf{X}_i, \mathbf{E}_i)$
13 Define fusion network $f_{\text{fuse}} : \mathbb{R}^{n \cdot d'} \to \mathbb{R}$ for output aggregation
14 Define $\mathcal{O} \leftarrow \texttt{AdamW}(\cup_i f_i \cup f_{\text{fuse}}, \eta)$, $\mathcal{L} \leftarrow \texttt{SmoothL1Loss}()$
15 **Step C: Joint Agent-Fusion Training**
16 **for** *epoch* $= 1$ *to* T **do**
17 **foreach** *agent* f_i **do**
18 Compute embedding $\mathbf{Z}_i \leftarrow f_i(\mathbf{X}_i, \mathbf{E}_i)$
19 Concatenate: $\mathbf{Z} \leftarrow [\mathbf{Z}_1 \| \mathbf{Z}_2 \| \ldots \| \mathbf{Z}_n] \in \mathbb{R}^{|\mathcal{V}| \times nd'}$
20 Predict: $\hat{\mathbf{y}} \leftarrow f_{\text{fuse}}(\mathbf{Z})$
21 Compute loss $\mathcal{L}_{\text{cur}} \leftarrow \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$; update all models
22 Apply early stopping based on $\mathcal{L}_{\text{cur}}$
23 **Step D: Inference and Deployment**
24 Reload best checkpoints $\{f_i^*\}, f_{\text{fuse}}^*$
25 **return** Final predictor $f^* = f_{\text{fuse}}^*([f_1^*(\cdot), \ldots, f_n^*(\cdot)])$

stopping based on the best observed loss $\mathcal{L}_{\text{cur}}$ (line 22) to ensure generalization. Upon convergence, the best-performing checkpoints $\{f_1^*, \ldots, f_n^*, f_{\text{fuse}}^*\}$ are reloaded (line 24) and the final prediction function is defined as:

$$f^*(\cdot) = f_{\text{fuse}}^* \left([f_1^*(\cdot), \ldots, f_n^*(\cdot)] \right)$$

This design enables flexible, modular reasoning across typed graph partitions and supports domain-independent deployment in fields such as agriculture, health, or economics.

3) Post-Hoc Symbolic Explanation Module: To enhance model interpretability, we implement a post-hoc symbolic explanation layer that operates independently of the prediction pipeline. After the multi-agent GNN produces a fertilizer recommendation, the system identifies the top-k contributing input features from each domain-specific agent (e.g., SoilAgent, WeatherAgent, CropAgent) based on their absolute values.

These features are interpreted using expert-defined thresholds to produce semantically meaningful labels. For instance, a pH value below 6.5 is categorized as "acidic," temperature below 18°C as "cold," and low nitrate concentrations as "low nitrogen." This interpretation process is rule-based and designed to reflect domain knowledge.

The resulting labels from each agent are then composed into a structured explanation template. For example: *"SoilAgent detects acidic conditions and low nitrogen. WeatherAgent observes dry weather. CropAgent indicates early growth stage. As a result, the system recommends 47.2 kg/ha of fertilizer."*

V. Experiments

A. Dataset Preparation

To construct the Knowledge Graph (KG) for our fertilizer recommendation framework, we gathered and integrated heterogeneous agricultural datasets representing four key domains: soil characteristics, crop information, fertilizer properties, and environmental conditions. Our study focuses on Isanti City, Minnesota, leveraging publicly available datasets from the USDA, Visual Crossing, and the EDI Data Portal. Each dataset underwent preprocessing, normalization, and semantic alignment before integration into the KG using Neo4j.

- **Soil Data:** Sourced from the USDA Web Soil Survey [24], this dataset includes nutrient levels (e.g., nitrogen, phosphorus, potassium), pH, moisture retention, and organic matter content. These attributes were used to construct soil nodes and inform nutrient-related relationships. Categorical values were one-hot encoded, and numerical values were scaled using MinMax normalization.
- **Fertilizer Data:** Obtained from the Environmental Data Initiative (EDI) repository [25], this dataset provides information on fertilizer types, nutrient composition, environmental impact, and application recommendations. It was used to create fertilizer nodes and define their interactions with both crop and soil entities.
- **Crop Data:** From the USDA National Agricultural Statistics Service (NASS) [26], this dataset includes crop types, growth cycles, yield expectations, and nutrient needs. Crop nodes were enriched with these attributes and linked to soil and fertilizer nodes through compatibility and demand relationships.
- **Weather Data:** Using Visual Crossing's Weather Query Builder [27], we accessed historical and real-time meteorological data, including temperature, humidity, and rainfall. These features formed the basis for weather nodes, enabling dynamic relationships with crop and soil nodes.

B. Knowledge Graph Schema

The constructed Knowledge Graph (KG) serves as a structured and semantically enriched representation of the heterogeneous agricultural data curated for this study. It comprises 3,792 nodes and 7,708 edges, reflecting the complex interdependencies among soil, crop, fertilizer, and environmental factors relevant to precision agriculture. Nodes are labeled according to domain-specific entities and attributes, including soil characteristics (e.g., *Soil Texture*, *Organic Matter (%)*), environmental conditions (e.g., *Humidity*, *Soil Moisture (%)*), crop parameters (e.g., *Expected Yield (tons/ha)*, *Growth Percentage (%)*), and

fertilizer profiles (e.g., *Fertilizer (kg/ha)*, *Phosphorous (ppm P in soil)*).

Each node encapsulates a feature vector constructed from normalized numerical attributes and encoded categorical variables, derived from authoritative data sources including the USDA Web Soil Survey and NASS [24], [26], Visual Crossing [27], and the Environmental Data Initiative (EDI) repository [25]. In addition to these features, metadata fields such as `id`, `source_sheet_rows`, and `source_files` were embedded to support traceability, provenance, and auditability.

The KG models 31 unique relationship types that capture both structural and semantic connections across entities. These relationships fall into three main categories:

- Impact relationships: such as *affectsExpectedYield*, *affectsGrowthPercentage*, and *hasFertilizerDecision*, which describe direct influence among variables.
- Compositional and structural links: including *hasDepth*, *hasCEC*, *hasClassification*, and *hasDrainageClass*, which define the physical and chemical properties of agricultural elements.
- Associative relationships: like *correlatesWith*, *associatedWith*, and *alters*, which support exploratory and inferential queries beyond explicitly stated dependencies.

These interconnected relationships enable multi-hop reasoning and semantic path traversal—capabilities essential for downstream tasks such as graph-based learning and agent-driven optimization. The graph was implemented using Neo4j (v5.24.0 Enterprise Edition), with data ingested through the Cypher query language and managed within a clustered database environment. This knowledge representation forms the backbone of our KG-MASFO architecture, supporting both predictive modeling and real-time decision-making within the proposed Multi-Agent System (MAS).

C. Evaluation Metrics

To assess model performance and learning behavior, we employ a suite of regression metrics and training diagnostics. During training, we use Smooth L1 Loss (Huber Loss) [22] as the optimization objective, due to its robustness to outliers and balanced gradient behavior.

After training, we evaluate performance on the test set using:

- **Mean Squared Error (MSE):** Measures the average squared difference between predictions and ground truth. Lower values indicate better prediction fidelity.
- **R-Squared (R^2):** Represents the proportion of variance explained by the model. Higher R^2 indicates stronger generalization and fit.

To ensure generalization, we apply early stopping [28] by monitoring validation loss. Additionally, training loss is visualized over epochs to inspect convergence stability, detect potential overfitting, and compare learning dynamics across model configurations.

D. Baselines

To evaluate the performance and contributions of our proposed KG-MASFO framework, we compare it against two internal baselines that represent simplified versions of the system. This progressive evaluation strategy allows us to isolate the effect of key architectural components such as agent modularity and multi-agent coordination.

- **GNN_GAT:** This baseline consists of a standard Graph Attention Network (GAT) model [29] trained end-to-end on the constructed Knowledge Graph without any agent modularity. It directly consumes node features and edge connections to predict fertilizer recommendations. This serves as the lower-bound benchmark for graph-based modeling without specialization or coordination.
- **Single-Agent GAT:** In this configuration, a single specialized GAT model is trained to focus on a specific subset of features (e.g., crop or soil information). It reflects a modular approach without inter-agent interaction or decision fusion. This setup allows us to evaluate whether isolated learning agents can improve performance by narrowing focus on domain-relevant variables.
- **Multi-Agent System:** This is the full KG-MASFO pipeline, in which multiple specialized GAT-based agents (e.g., soil, crop, weather) independently process inputs from the Knowledge Graph. Their embeddings are then fused and passed to a meta-agent, which synthesizes the predictions. This architecture is designed to promote both specialization and collaboration, and is expected to yield improved performance and interpretability.

This tiered baseline strategy enables a fair and interpretable comparison of predictive accuracy, training stability, and generalization across increasingly sophisticated levels of model design.

E. Experimental Results

To evaluate the predictive performance and learning dynamics of our proposed KG-MASFO framework, we compare three model configurations: (1) a baseline Graph Attention Network (GNN_GAT) trained on the full Knowledge Graph without modularity, (2) a Single-Agent GAT model trained on specialized features, and (3) our Multi-Agent GAT architecture, which fuses outputs from independently trained agents via a meta-agent.

a) Quantitative Comparison.: Table I summarizes the final Mean Squared Error (MSE), R-squared (R^2) scores, and total training epochs for each phase. To ensure fair comparison, all models were trained for 251 epochs. The GNN_GAT baseline achieved an MSE of 0.0074 and an R^2 of 0.9025. The single-agent configuration improved both metrics, achieving an MSE of 0.0058 and R^2 of 0.9515. Our multi-agent architecture outperformed both, reaching a significantly lower MSE of 0.0020 and a higher R^2 of 0.9739, indicating superior predictive accuracy and model fit.

b) Training Dynamics.: Figure 3 shows the training loss curves over 251 epochs. The GNN_GAT model converges slowly and plateaus at a higher loss, indicating limited learning capacity in the absence of modular specialization. The single-agent model converges rapidly and stabilizes early, demonstrating improved generalization from focused feature learning. The

TABLE I

PERFORMANCE COMPARISON ACROSS MODEL PHASES. THE PROPOSED MULTI-AGENT KG-MASFO FRAMEWORK ACHIEVES THE LOWEST MSE AND HIGHEST R^2 SCORE, DEMONSTRATING SUPERIOR PREDICTION ACCURACY AND GENERALIZATION.

Model Variant	MSE ↓	R^2 Score ↑
GNN_GAT	0.0074	0.9025
Single-Agent GAT	0.0058	0.9515
KG-MASFO	**0.0020**	**0.9739**

multi-agent model exhibits both rapid convergence and long-term stability, maintaining the lowest loss across all epochs. This suggests that its architecture better captures multi-factor dependencies and benefits from collaborative reasoning among specialized agents.

c) Discussion.: These experimental results validate the effectiveness of the KG-MASFO framework. The inclusion of modular agents, each trained on domain-specific features, and a meta-agent that learns to synthesize predictions, enables more accurate and context-aware fertilizer recommendations. The combination of improved MSE, elevated R^2, and stable training behavior highlights the system's ability to generalize across heterogeneous data types and environmental conditions. This modular design also opens avenues for explainability and real-time decision support in future deployments.

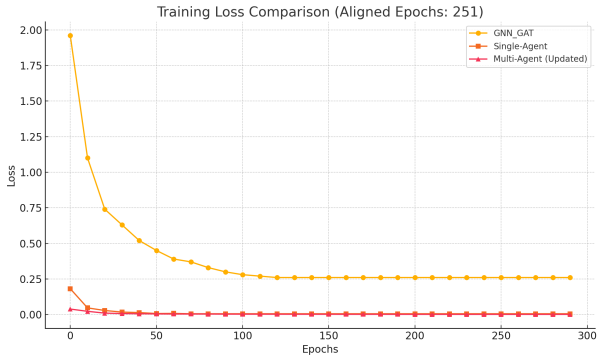

Fig. 3. Training loss curves over 251 aligned epochs for all model phases. The multi-agent model achieves the lowest loss and maintains stability across training, demonstrating superior learning capacity and convergence behavior.

F. Evaluation of Symbolic Reasoning Module

To evaluate the interpretability of our symbolic post-hoc explanation module, we present a representative use case that demonstrates how the system transforms numeric predictions into actionable, domain-informed justifications. The explanation module identifies salient input features from each agent's subgraph (e.g., soil, weather, crop), maps them to qualitative descriptors using expert rules, and assembles them into human-readable reasoning templates.

Explanation Scenario: Fertilizer Prediction under Adverse Conditions

Input Features:

- *Soil:* pH = 5.8, NO_3 = 6.0 mg/kg, Calcium = 8.5 ppm
- *Weather:* Temperature = 11C, Humidity = 22%, Precipitation = 3.0 mm
- *Crop:* Growth Progress = 12%, Pesticide Applied = 0.0 L/ha

System Interaction:

- *User:* What is the recommended fertilizer dose?
- *System:* The predicted fertilizer application is `39.7 kg/ha`.
- *User:* Why this value?
- *System:* The soil is acidic (pH 5.8), nitrogen-deficient (NO_3 = 6.0 mg/kg), and low in calcium. Weather conditions are cold (11C), dry (22% humidity), and rain-scarce (3.0 mm precipitation). The crop is in an early development stage (12%). These combined stress signals justify a moderate fertilizer dose to stimulate growth under constrained environmental conditions.
- *User:* What type of fertilizer does this suggest?
- *System:* A balanced NPK fertilizer containing moderate nitrogen levels—appropriate for crops at an early stage, especially in low-nutrient and low-temperature settings.

a) Discussion.: This example highlights the system's ability to produce structured, interpretable justifications grounded entirely in input features and domain-specific rules. Unlike black-box explainability methods, our symbolic module operates deterministically, ensuring transparency, reproducibility, and alignment with expert reasoning. The modular design also supports real-time user interaction and domain expert validation, making it suitable for decision support and agricultural informatics deployments.

VI. CONCLUSIONS

In this study, we proposed KG-MASFO, a Knowledge Graph-based Multi-Agent System for Fertilizer Optimization that integrates graph neural networks, agent-based reasoning, and structured agricultural data. Our approach leverages domain-aware modeling through a unified Neo4j knowledge graph, enabling predictive learning over soil, crop, fertilizer, and environmental interdependencies.

Through a tiered evaluation framework, we compared three progressively advanced model configurations: a baseline GNN_GAT model, a single-agent GAT architecture, and our proposed multi-agent pipeline. Experimental results demonstrated that the multi-agent approach consistently outperformed both baselines, achieving a final MSE of 0.0020 and an R^2 score of 0.9739. This confirms that incorporating specialized agents and a meta-agent fusion mechanism significantly enhances predictive accuracy and learning stability.

The system design supports scalability, interpretability, and adaptability—key requirements for real-world agricultural applications. Additionally, the use of attention mechanisms and structured feature encoding enables transparent, explainable AI-based recommendations. By integrating real-time environmental data and domain knowledge into a unified reasoning system, KG-MASFO addresses the limitations of static and rule-based fertilization strategies.

Future work will focus on extending the agent framework with game-theoretic coordination strategies, incorporating edge computing for real-time field deployment, and enhancing explainability via SHAP analysis and visual feedback. The results of this research not only advance the field of precision agriculture but also offer a generalizable methodology for multi-agent decision-making over knowledge-enriched graph representations in other domains.

References

[1] A. Patel, R. Sharma, and S. Verma, "Ai-powered precision agriculture: A review," *Journal of Agricultural Science*, 2021.

[2] A. O. Adewusi, O. F. Asuzu, T. Olorunsogo, C. Iwuanyanwu, E. Adaga, and D. O. Daraojimba, "Ai in precision agriculture: A review of technologies for sustainable farming practices," *World Journal of Advanced Research and Reviews*, vol. 21, no. 1, pp. 2276–2285, 2024.

[3] L. I. Hankovszky *et al.*, "Decision support system for sustainable nitrogen fertilization," *Precision Agriculture*, vol. 16, pp. 675–688, 2015.

[4] T. Talaviya, D. Shah, N. Patel, H. Yagnik, and M. Shah, "Implementation of artificial intelligence in agriculture for optimisation of irrigation and application of pesticides and herbicides," *Artificial Intelligence in Agriculture*, vol. 4, pp. 58–73, 2020.

[5] N. N. Rabalais, R. E. Turner, and R. J. Diaz, "Global change and eutrophication of coastal waters," *Limnology and Oceanography*, vol. 54, no. 6part2, pp. 1712–1721, 2019.

[6] STLPR, "Conservation and progress slashing gulf's 'dead zone' lags," https://www.stlpr.org/health-science-environment/2024-06-24/missouri-gulf-of-mexico-fertilizer-runoff, 2024, accessed: 2024-06-24.

[7] R. Yan, P. An, X. Meng, Y. Li, D. Li, F. Xu, and D. Dang, "A knowledge graph for crop diseases and pests in china," *Nature*, vol. 590, no. 7845, pp. 248–252, 2025.

[8] X. Zhao, Y. Liu, and Z. Wang, "Graph-based ai in bioinformatics," *Bioinformatics Journal*, 2020.

[9] J. Li, M. Xu, L. Xiang, D. Chen, W. Zhuang, X. Yin, and Z. Li, "Large language models and foundation models in smart agriculture: Basics, opportunities, and challenges," *Journal of AI in Agriculture*, vol. 8, no. 2, pp. 112–130.

[10] Kellogg Biological Station LTER, "Protocol - nitrogen deposition study – fertilizer treatments," https://lter.kbs.msu.edu/protocols/121, 2024.

[11] Penn State, "Managing runoff to reduce the dead zone," https://www.e-education.psu.edu/geog3/node/1114, accessed: 2024-04-03.

[12] NPR, "To heal the gulf of mexico's dead zone, we have to look north to midwest farms," https://www.npr.org/2024/08/15/nx-s1-5053072/to-heal-the-gulf-of-mexicos-dead-zone-we-have-to-look-north-to-midwest-farms, 2024, accessed: 2024-08-15.

[13] Kellogg Biological Station LTER, "Kellogg biological station lter," https://lternet.edu/wp-content/uploads/2019/12/KBS_brief_FINAL.pdf, 2019.

[14] John Deere, "John deere launches exactshot™ at ces 2023," https://www.deere.com/en/news/all-news/2023jan03-exactshot/, 2023.

[15] A. O. Adewusi, O. F. Asuzu, T. Olorunsogo, C. Iwuanyanwu, E. Adaga, and D. O. Daraojimba, "Ai in precision agriculture: A review of technologies for sustainable farming practices," *World Journal of Advanced Research and Reviews*, vol. 21, no. 1, pp. 2276–2285, 2024.

[16] A. Mahajan, S. Hegde, E. Shay, D. Wu, and A. Prins, "Comparative analysis of multi-agent reinforcement learning policies for crop planning decision support," *arXiv preprint arXiv:2412.02057*, 2024.

[17] C. Zhao *et al.*, "A review of agricultural internet of things technology," *Computers and Electronics in Agriculture*, vol. 172, p. 105340, 2020.

[18] SmythOS, "How multi-agent systems are transforming agriculture: Applications in smart farming," https://smythos.com/ai-agents/multi-agent-systems/multi-agent-systems-in-agriculture/, 2024.

[19] C. Y. J. W. W. L. J. L. Ge Zhang, Zhijie Gao and H. Luo, "Kgansynergy: knowledge graph attention network for drug synergy prediction," *Briefings in Bioinformatics*, 2023, gATv2 is implemented in PyTorch Geometric, DGL, and TensorFlow GNN. [Online]. Available: https://academic.oup.com/bib/article/24/3/bbad167/7147878

[20] N. Inc., "Neo4j graph platform," https://neo4j.com/.

[21] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *International Conference on Learning Representations (ICLR)*, 2022, gATv2 is implemented in PyTorch Geometric, DGL, and TensorFlow GNN. [Online]. Available: https://doi.org/10.48550/arXiv.2105.14491

[22] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[23] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *International Conference on Learning Representations (ICLR)*, 2019, published as a conference paper at ICLR 2019. [Online]. Available: https://doi.org/10.48550/arXiv.2105.14491

[24] U.S. Department of Agriculture, "Web Soil Survey," https://websoilsurvey.nrcs.usda.gov/app/WebSoilSurvey.aspx, 2024, accessed: 2025-04-06.

[25] Environmental Data Initiative, "EDI Data Portal," https://portal.edirepository.org/nis/simpleSearch, 2024, accessed: 2025-04-06.

[26] USDA National Agricultural Statistics Service, "Minnesota State Statistics," https://www.nass.usda.gov/Statistics_by_State/Minnesota/, 2024, accessed: 2025-04-06.

[27] Visual Crossing Corporation, "Weather Data Query Builder," https://www.visualcrossing.com/weather-query-builder/, 2024, accessed: 2025-04-06.

[28] L. Prechelt, "Early stopping — but when?" in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 55–69.

[29] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.