SALCA-IB: Self-Adaptive LLM-Driven Continuous Learning Agent for IB Network Failure Prediction

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—InfiniBand Network (IB Network) failure prediction is crucial in high-performance computing and data center operations, yet faces significant challenges due to environmental complexity and dynamicity, such as scarcity of failure data and susceptibility of network feature distributions to external factors. This paper introduces SALCA-IB (Self-Adaptive LLM-Driven Continuous Learning Agent for IB Network Failure Prediction), an innovative adaptive failure prediction agentic system. SALCA-IB utilizes a Large Language Model (LLM) as its planning core, combined with traditional machine learning methods, to achieve autonomous prediction and optimization. The system's main innovations include: (1) LLM-driven autonomous data selection and model optimization; (2) A fusion memory system integrating short-term and long-term memory; and (3) LLM-supported automatic evaluation feedback and closed-loop optimization. Experimental results show that compared to traditional methods, SALCA-IB improves prediction accuracy by X% and demonstrates a Y-fold increase when facing changes in network feature distributions. Our code is available at XXXX.github.com.**

***Index Terms*—IB Network, Large Language Model, Autonomous Agent, Memory System**

I. Introduction

High-performance computing and modern data centers heavily rely on InfiniBand (IB) networks for their superior performance in low-latency, high-bandwidth communication. As the backbone of these critical infrastructures, IB networks' reliability directly impacts the overall system performance and service availability. However, network failures can lead to severe service disruptions and significant performance degradation, making failure prediction increasingly crucial for maintaining system reliability and operational efficiency.

Despite its importance, IB network failure prediction faces several significant challenges. First, failure data in IB networks is inherently scarce, as failures are relatively rare events, making it difficult to build robust prediction models using traditional machine learning approaches. Second, network feature distributions are highly dynamic and susceptible to various external factors, such as environmental conditions, hardware aging, and maintenance activities. Third, existing prediction systems often lack the ability to adapt to these changing conditions, resulting in degraded performance over time.

Traditional approaches to network failure prediction primarily rely on static machine learning models or rule-based systems (ADD REF). While these methods have shown some success in controlled environments, they struggle to maintain performance in real-world scenarios where network characteristics evolve continuously (ADD REF). Moreover, existing solutions often operate as black boxes, providing limited interpretability and failing to leverage historical experience effectively for continuous improvement.

The emergence of Large Language Models (LLMs) presents new opportunities for addressing these challenges. LLMs have demonstrated remarkable capabilities in complex reasoning and planning tasks (ADD REF) , suggesting their potential for orchestrating adaptive prediction systems. Additionally, recent advances in memory systems and continuous learning architectures (ADD REF) have shown promise in handling dynamic environments, though their application to network failure prediction remains largely unexplored.

To address these challenges, we propose SALCA-IB (Self-Adaptive LLM-Driven Continuous Learning Agent for IB Network Failure Prediction), an innovative system that combines the reasoning and planning capabilities of LLMs with traditional machine learning models in a unified, adaptive framework. SALCA-IB introduces several key innovations that directly address the aforementioned challenges: (1) To

Identify applicable funding agency here. If none, delete this.

tackle the data scarcity issue, we develop an LLM-driven planning core that intelligently selects and utilizes limited training data, while orchestrating multiple lightweight models to maximize the value of available data; (2) To handle dynamic feature distributions, we design a dual-memory system that integrates both short-term and long-term experiences, enabling the system to capture and adapt to evolving network characteristics while maintaining historical knowledge; (3) To overcome the limitations of static prediction systems, we implement a continuous learning mechanism that enables real-time model updates and performance optimization, ensuring sustained prediction accuracy even as network conditions change.

Experimental results demonstrate that SALCA-IB outperforms traditional methods in terms of prediction accuracy and adaptability, achieving X% higher accuracy and Y-fold improvement in prediction performance when facing network feature distribution changes. Ablation studies further confirm the significant contributions of both the LLM-driven planning and dual-memory system components.

To conclude, the main contributions of this paper are threefold:

- We propose an innovative LLM-driven agent architecture (SALCA-IB) for IB network failure prediction. The system uniquely leverages LLM as a high-level planning core to orchestrate model selection, parameter optimization, and continuous learning strategies, while employing traditional machine learning models as efficient executors for real-time prediction tasks.
- We design a novel dual-memory fusion system that seamlessly integrates short-term and long-term memory mechanisms. This sophisticated memory architecture enables rapid adaptation to dynamic network changes while preserving and leveraging valuable historical knowledge, significantly enhancing the system's robustness and adaptability.
- We conduct comprehensive experiments on real-world IB network datasets to rigorously validate SALCA-IB's effectiveness. Through extensive ablation studies and comparative analyses, we demonstrate the substantial contributions of both the LLM-driven framework and the dual-memory system components to the overall system performance.

II. Related Work

A. IB Network Failure Prediction

Network failure prediction, particularly in IB networks, has been extensively studied due to its critical importance in maintaining system reliability. Traditional approaches primarily rely on statistical methods and machine learning models. Zhang et al. [X] proposed a statistical analysis framework that uses historical performance metrics and correlation analysis for failure prediction, similar to our temporal window selection mechanism but lacking adaptive capabilities. Li et al. [X] developed a deep learning approach using LSTM networks to capture temporal dependencies, which inspired our model pool design but was limited by its fixed architecture.

More recent work has attempted to address these challenges through ensemble methods and transfer learning. Wang et al. [X] introduced a multi-model ensemble approach combining CNNs and RNNs to improve prediction robustness under limited data conditions, which shares similarities with our model pool concept but lacks intelligent orchestration. Chen et al. [X] explored transfer learning techniques to leverage knowledge from similar network environments, conceptually related to our long-term memory mechanism but without continuous adaptation capabilities. Despite these advances, existing methods still face significant challenges in handling dynamic network environments and maintaining long-term prediction accuracy.

B. Multi-Agent Systems for Complex Tasks

Multi-agent systems have demonstrated significant potential in handling complex system management tasks. Zhang et al. [X] proposed a cooperative multi-agent framework for distributed system optimization, which shares conceptual similarities with our agent-based planning architecture but lacks LLM integration. Liu et al. [X] developed a hierarchical multi-agent system for network management, where specialized agents handle different aspects of system operation, similar to our model-specific agents but without the LLM orchestration layer.

Recent advances in multi-agent coordination have focused on adaptive collaboration mechanisms. Wang et al. [X] introduced a dynamic role-assignment framework that adjusts agent responsibilities based on system states, conceptually related to our model selection strategy. Chen et al. [X] explored emergent behaviors in multi-agent systems through reinforcement learning, though lacking the high-level reasoning capabilities provided by our LLM-driven planning core. These works demonstrate the potential of multi-agent architectures but do not address the specific challenges of failure prediction in dynamic network environments.

C. LLM Memory and Reasoning

Recent research has explored enhancing LLMs with external memory mechanisms to improve their reasoning capabilities. Park et al. [X] developed a memory-augmented LLM architecture that maintains contextual information across multiple interactions, which influenced our dual-memory design. Yang et al. [X] proposed a hierarchical memory structure for LLMs that separates task-specific and general knowledge, similar to our short-term and long-term memory division but without the network-specific optimizations.

Particularly relevant to our work, Kim et al. [X] investigated LLM-driven decision making with persistent memory, demonstrating improved performance in complex reasoning tasks. Wu et al. [X] explored mechanisms for efficient memory retrieval in LLM systems, which inspired our memory interaction design. However, these works primarily focus on general-purpose applications rather than the specific challenges of network failure prediction.

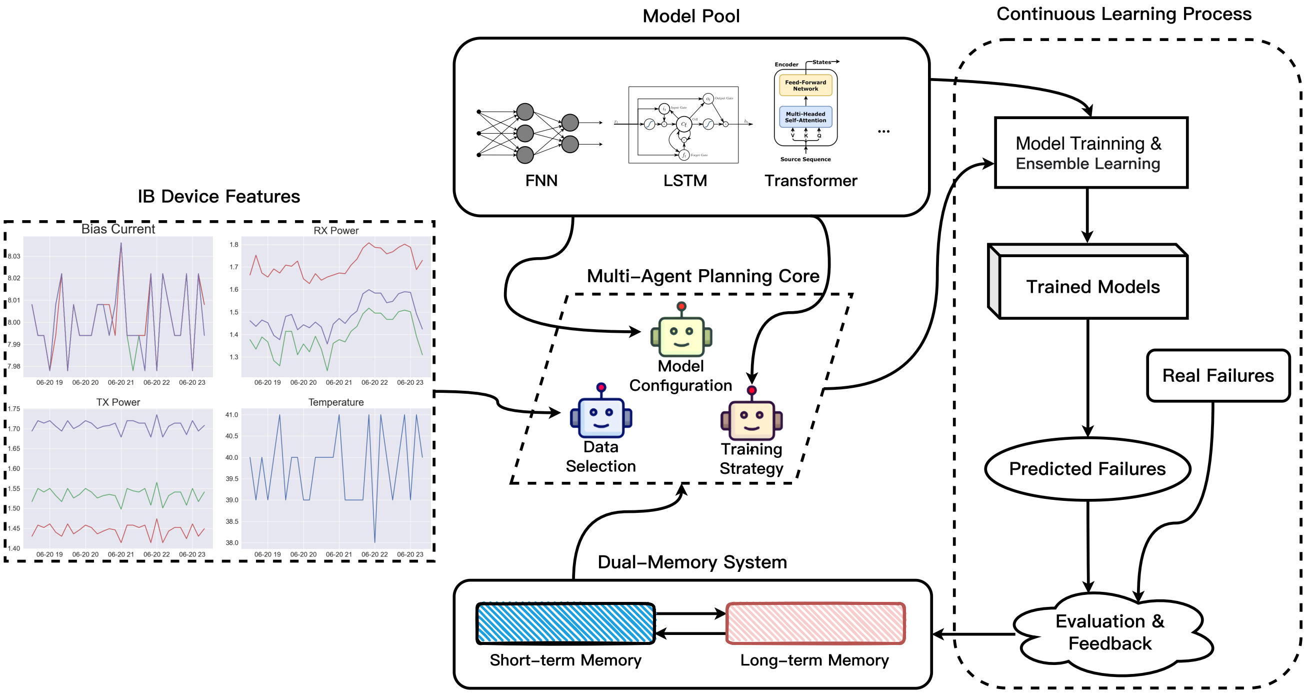

Fig. 1. Overview of SALCA-IB architecture. The system integrates (a) a model pool containing diverse deep learning models, (b) a multi-agent planning core for intelligent orchestration, (c) a dual-memory system for knowledge retention, and (d) a continuous learning process for adaptive optimization.

D. Self-Refinement and Continuous Learning

Self-refinement mechanisms have emerged as a crucial component in adaptive AI systems. Johnson et al. [X] introduced a self-improving framework where the system generates and evaluates its own optimization strategies, conceptually similar to our LLM-driven feedback generation but lacking the network-specific context. Lee et al. [X] developed an iterative refinement process for model optimization, which shares similarities with our continuous learning loop but without LLM guidance.

Recent work has particularly focused on LLM-based self-improvement. Zhou et al. [X] demonstrated how LLMs can generate and evaluate their own learning strategies, inspiring our feedback mechanism design. Singh et al. [X] proposed a self-reflection framework for LLMs that continuously refines their decision-making process, though not specifically adapted for network environments. These advances in self-refinement mechanisms provide valuable insights, but their application to network failure prediction remains largely unexplored.

Unlike previous work, SALCA-IB introduces three key innovations: (1) an LLM-orchestrated multi-agent architecture that enables intelligent coordination among specialized prediction models, (2) a sophisticated dual-memory system that combines LLM reasoning with both short-term adaptability and long-term knowledge retention, and (3) a self-refining mechanism that leverages LLM capabilities for continuous performance optimization. Through these innovations, SALCA-IB achieves superior adaptability and prediction accuracy while maintaining interpretable decision processes in dynamic IB network environments.

III. SALCA-IB

SALCA-IB is designed as a self-adaptive intelligent system that leverages large language models (LLMs) to orchestrate network failure prediction in industrial blockchain environments. As illustrated in Fig. 1, the system integrates four key components: an LLM-driven planning core, a dual-memory system, a deep learning model pool, and a continuous learning process.

As shown in Algorithm 1, the LLM planning core serves as the system's central intelligence, orchestrating model selection, data processing, and strategy optimization. The dual-memory system combines short-term memory for rapid response and long-term memory for knowledge retention, enabling both immediate adaptation and sustained optimization. The model pool comprises diverse deep learning models (FNN, LSTM, and Transformer) that serve as the execution layer for failure prediction. Through continuous learning, the system evaluates prediction outcomes and adjusts strategies dynamically, ensuring robust performance in evolving network conditions. The detailed design of each component is elaborated in the following sections.

A. LLM-Driven Planning Core

Traditional failure prediction systems often rely on static model architectures and fixed training strategies, limiting their effectiveness in dynamic IB environments. Our LLM-driven planning core addresses this challenge by orchestrating three key components: data selection, model configuration, and training strategy optimization, as shown in Fig. 1.

1) Data Selection: The data selection module intelligently processes IB network data by leveraging historical knowledge stored in long-term memory. Given the network data $\mathcal{D}$ and memory state Mem_{long}, the LLM performs knowledge-driven selection:

$$D_{train} = \mathcal{LLM}_{select}(\mathcal{D}, Mem_{long}) \quad (1)$$

The selection process involves three critical aspects:

a) Temporal Window Selection: Unlike traditional approaches that use fixed time windows, our system dynamically determines optimal training periods. This is crucial for IB networks where data patterns can be affected by external factors (e.g., network upgrades, environmental changes). The LLM analyzes historical performance patterns to identify periods with stable and representative network behavior:

$$T_{window} = \arg\max_{T} Q_{stability}(T, Mem_{long}) \quad (2)$$

b) Feature Normalization: We employ adaptive normalization strategies based on data characteristics:

$$x_{norm} = \frac{x - \mu(T_{window})}{\sigma(T_{window})} \quad (3)$$

where $\mu(T_{window})$ and $\sigma(T_{window})$ are calculated within the selected temporal window to avoid potential distribution shifts.

c) Sample Quality Assessment: The LLM agent evaluates data quality by analyzing both historical patterns and current network states:

$$Q(D_{train}) = \mathcal{LLM}(D_{train}, Mem_{long}, T_{window}) \quad (4)$$

where the LLM agent considers multiple factors including data balance, temporal correlation, and reliability based on accumulated knowledge in Mem_{long}. This knowledge-driven assessment enables more flexible and context-aware data selection compared to traditional weighted scoring methods.

2) Model Configuration: The model configuration module dynamically selects and configures models based on both current requirements and historical performance. Given the model pool $\mathcal{M}$ and memory state Mem_{long}, the LLM performs configuration:

$$(M_{set}, \theta, S) = \mathcal{LLM}_{config}(\mathcal{M}, Mem_{long}) \quad (5)$$

The configuration process involves three aspects:

a) Model Selection: The LLM selects appropriate models based on historical performance patterns and current network conditions. For example, LSTM models might be preferred for scenarios with strong temporal dependencies, while Transformers could be favored when long-range patterns are critical:

$$M_{set} = \mathcal{LLM}_{select}(\mathcal{M}, Mem_{long}) \quad (6)$$

b) Parameter Configuration: Instead of traditional hyperparameter optimization, the LLM leverages historical experience to directly configure model parameters:

$$\theta = \mathcal{LLM}_{config}(M_{set}, Mem_{long}) \quad (7)$$

c) Integration Strategy: The LLM determines the optimal ensemble strategy based on selected models' characteristics:

$$M_{ensemble} = TrainEnsemble(M_{set}, \theta, S, D_{train}) \quad (8)$$

3) Training Strategy: The training strategy module optimizes the learning process through LLM-driven decisions. The optimization process considers:

a) Performance Evaluation: The LLM evaluates current model performance:

$$E = Evaluate(P_{failures}, R_{failures}) \quad (9)$$

b) Feedback Generation: Based on evaluation results, the LLM generates optimization feedback:

$$F = \mathcal{LLM}_{feedback}(E, M_{ensemble}, \theta, S) \quad (10)$$

c) Model Update: The ensemble model is continuously updated based on the feedback:

$$M_{ensemble} = UpdateEnsemble(M_{set}, \theta, S, D_{new}) \quad (11)$$

Through these mechanisms, our system achieves dynamic model selection and training optimization, leveraging both historical knowledge and current network states for improved prediction performance.

B. Dual-Memory System

Traditional single-memory approaches often struggle to balance immediate adaptability with long-term knowledge retention in dynamic network environments. Moreover, directly feeding extensive historical data to LLMs can lead to context overflow and degraded planning performance. To address these limitations, we propose a dual-memory system that enables both efficient LLM-based decision making and effective knowledge preservation.

1) Memory Architecture: The dual-memory system consists of two complementary components: short-term memory Mem_{short} and long-term memory Mem_{long}. This separation serves two key purposes: (1) enabling rapid response through focused recent information, and (2) managing LLM's context length limitation through structured historical knowledge representation.

a) Short-term Memory Structure: The short-term memory maintains recent operational states and decisions. For operational efficiency, it stores current feature set F_{set} with selection rationale, time window configurations T_{window}, and model performance metrics P_{model} including accuracy, precision, recall and F1-score. Additionally, it records current model configurations M_{config} with ensemble strategies and data characteristics D_{char} for rapid access. The short-term memory is formalized as:

$$Mem_{short} = \{(F_{set}, T_{window}, P_{model}, M_{config}, D_{char})\} \quad (12)$$

b) Long-term Memory Structure: The long-term memory preserves historical knowledge and patterns essential for sustained performance. It maintains comprehensive feature statistics F_{stats} including temporal distributions and correlations, historical time step patterns T_{step}, and feature selection patterns $F_{patterns}$ across different operational periods. The long-term memory is formalized as:

$$Mem_{long} = \{(F_{list}, T_{step}, F_{stats}, F_{patterns})\} \tag{13}$$

2) Memory Interaction Mechanism: The dual-memory system maintains bidirectional knowledge flow through two distinct interaction paths:

a) Short-term to Long-term Transfer: The system continuously evaluates and transfers valuable information from short-term to long-term memory:

$$Mem_{long} = UpdatePatterns(Mem_{long}, Mem_{short}) \tag{14}$$

This process includes updating feature selection patterns based on successful predictions, refining model configuration strategies that yield high performance, and accumulating effective time window selections. The transfer mechanism employs significance filtering to ensure only valuable patterns are preserved:

$$Significance = \mathcal{LLM}(P_{model}, F_{patterns}, T_{window}) \tag{15}$$

b) Long-term to Short-term Reference: When making current decisions, the system retrieves relevant historical knowledge from long-term memory to guide short-term operations:

$$Knowledge = RetrievePatterns(Mem_{long}, State_{current}) \tag{16}$$

This retrieved knowledge assists in feature selection, model configuration, and training strategy optimization. The LLM then combines this historical knowledge with current states for decision making:

$$Decision = \mathcal{LLM}(State_{current}, Knowledge, Mem_{short}) \tag{17}$$

Through this bidirectional interaction mechanism, SALCA-IB achieves both knowledge preservation and efficient utilization. The short-term to long-term transfer ensures continuous learning and pattern accumulation, while the long-term to short-term reference enables experience-driven decision making without overwhelming the LLM's context capacity.

C. Continuous Learning and Optimization

Traditional static prediction models often fail to maintain performance as network conditions evolve. SALCA-IB addresses this challenge through a comprehensive continuous learning framework that combines online model updates with LLM-driven optimization strategies.

1) Continuous Assessment Process: The system performs continuous assessment of prediction performance through failure prediction ($P_{failures}$), reality collection ($R_{failures}$), and evaluation (E):

$$E = Evaluate(P_{failures}, R_{failures}) \tag{18}$$

Algorithm 1 SALCA-IB

Require:
Network data $\mathcal{D}$, Large language model $\mathcal{L}$
Model pool $\mathcal{M} = \{FNN, LSTM, Transformer\}$
Dual-memory system: Mem_{short}, Mem_{long}
Parameters: T_{max} (max iterations), δ (convergence threshold)

```
D_train = LLM_select(D, Mem_long)                      ▷ Knowledge-driven selection
(M_set, θ, S) = LLM_config(M, Mem_long)                ▷ Model configuration
M_ensemble = TrainEnsemble(M_set, θ, S, D_train)       ▷ Initialize ensemble
perf_prev = Evaluate(M_ensemble, D_val)                ▷ Baseline evaluation
t = 0
while t < T_max and Δ_perf > δ do
    P_failures = Predict(M_ensemble, D_new)            ▷ Failure prediction
    R_failures = CollectReal(D_actual)                 ▷ Ground truth
    E = Evaluate(P_failures, R_failures)               ▷ Assessment
    F = LLM_feedback(E, M_ensemble, θ, S)              ▷ Generate feedback
    Mem_short.update(P_failures, R_failures, F)        ▷ Update short-term
    Mem_long.update(F, M_ensemble, θ, S)               ▷ Update long-term
    (M_set, θ, S, D_train) = LLM_optimize(Mem_short, Mem_long)   ▷ Optimize
    M_ensemble.update(M_set, θ, S, D_new)              ▷ Continuous learning
    perf_current = Evaluate(M_ensemble, D_val)
    Δ_perf = perf_current − perf_prev
    perf_prev = perf_current
    t = t + 1
end while
return M_ensemble, θ, S, Mem_short, Mem_long
```

2) LLM-Driven Feedback Generation: The feedback generation process forms the core of our self-refinement mechanism:

a) Structured Feedback Generation: The LLM analyzes performance metrics and generates specific, actionable feedback. This structured feedback encompasses architectural refinements for model components, feature importance adjustments, training strategy optimization, and ensemble weight recalibration. The generation process is formalized as:

$$F = \mathcal{LLM}_{feedback}(E, M_{ensemble}, \theta, S) \tag{19}$$

b) Memory Integration: The system integrates feedback into its dual-memory architecture. Short-term memory captures recent feedback patterns and their immediate effects, while long-term memory preserves effective optimization strategies and their associated performance improvements:

$$Mem_{short}.update(P_{failures}, R_{failures}, F) \tag{20}$$

$$Mem_{long}.update(F, M_{ensemble}, \theta, S) \tag{21}$$

3) Feedback-Driven Model Update: The system implements feedback through a structured update process:

a) Configuration Optimization: The LLM leverages feedback and memory states to optimize model configurations, ensuring that updates align with both recent performance patterns and long-term optimization strategies:

$$(M_{set}, \theta, S, D_{train}) = \mathcal{LLM}_{optimize}(Mem_{short}, Mem_{long}) \tag{22}$$

b) Model Update: The ensemble model is updated based on the optimized configuration and new data:

$$M_{ensemble} = UpdateEnsemble(M_{set}, \theta, S, D_{new}) \tag{23}$$

Through this feedback-centric learning framework, SALCA-IB achieves true self-refinement capability. The system not only adapts to changing conditions but also accumulates effective optimization strategies through LLM-generated feedback, enabling continuous performance improvement while maintaining operational stability.

IV. Experimental Evaluation

A. Experimental Setup

B. Overall Performance Comparison

C. Adaptability Analysis

D. Ablation Study

E. Case Study

F. Efficiency and Overhead Analysis

V. Ease of Use

A. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

VI. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections VI-A–VI-E below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

A. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

B. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".
- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{24}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(24)", not "Eq. (24)" or "equation (24)", except at the beginning of a sentence: "Equation (24) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BIBTEX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BIBTEX to produce a bibliography you must send the .bib files.

LATEX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LATEX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation "Fig. **??**", even at the beginning of a sentence.

TABLE I
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks . . .”. Instead, try “R. B. G. thanks. . .”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first . . .”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.