NLP based Deep Classifier for Metalevel Tracking

Yiming Zhang
Cornell University
Ithaca, USA
yz2926@cornell.edu

Vikram Krishnamurthy
Cornell University
Ithaca, USA
vikramk@cornell.edu

Shashwat Jain
Cornell University
Ithaca, USA
sj474@cornell.edu

***Abstract*—This paper constructs metalevel tracking algorithm for classifying the trajectory of a moving target over long time horizons, so as to determine the target's intent. The metalevel tracker operates seamlessly above the classical radar tracking algorithm. To construct Natural Language Processing (NLP) based generative models for complex syntactic patterns of target trajectories, we employ Stochastic Context Sensitive Grammar (SCSG). However, Bayesian inference of SCSG is NP-hard; so we construct deep neural networks that exploit the SCSG parse-tree structure. We show, via extensive numerical examples, that the resulting grammar-aware deep classifier performs substantially more accurately in classifying target trajectories, compared to other traditional trajectory classification methods. Our approach highlights the potential of carefully constructed parse-tree aware deep classifiers for accurate and interpretable trajectory intent analysis for metalevel tracking.**

***Index Terms*—Stochastic Context Sensitive Grammar, Syntactic Trajectory Patterns, Metalevel Tracking, Deep Classifier**

I. Introduction

Classical radar-based target tracking assumes a Markov state space model for the target kinematics. Such models are useful on short timescales (on the order of several seconds), and numerous target tracking algorithms based on such Markov models have been developed in the literature [20], [23]. This paper is motivated by metalevel tracking on longer timescales (on the order of several minutes). In metalevel tracking, one is interested in devising automated procedures that assist a human analyst to interpret the tracks obtained from a conventional tracking algorithm. On such longer timescales, real-world targets are driven by a premeditated intent. The intent of a target manifests in the shape of the target trajectory. In this paper, the the trajectory shape is modeled using a sophisticated natural language based method called a stochastic context sensitive grammar (SCSG). SCSGs serve as generative models for complex spatio-syntactic patterns [1], [10], [16], [24].

Fig. 1 illustrates the main idea of this paper. Suppose a target moves in a rectangle (of arbitrary size) around a building on a road grid, such intent is suspicious. Given the 'dots' from the classical tracker, how to 'join' these dots to estimate the target's trajectory? A Markov chain cannot exclusively generate a rectangle of arbitrary size (due to the dependency of the two sides being arbitrary but equal). The pumping lemma from formal language theory [13] shows that an SCSG serves as a generative model in that it can exclusively generate rectangles where the two dimensions are positive integer-valued random variables. Several other complex trajectories can be generated

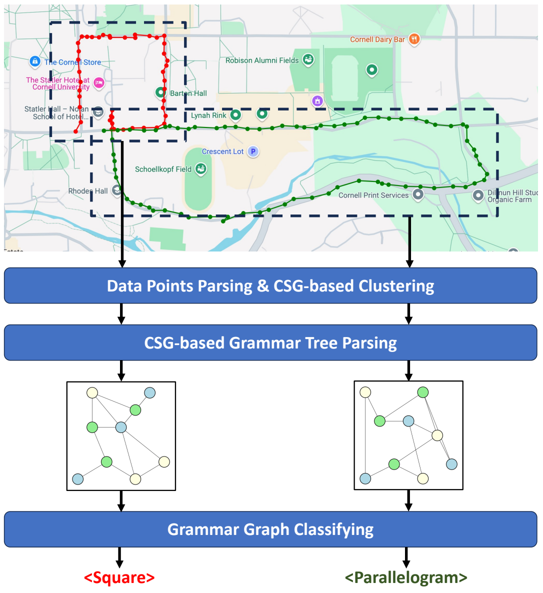

The Cornell Store
The Statler Hotel at Cornell University
Statler Hall – Nolan School of Hotel...
Barton Hall
Robison Alumni Fields
Cornell Dairy Bar
Lynah Rink
Crescent Lot
Schoellkopf Field
Rhodes Hall
Cornell Print Services
Dilmun Hill Student Organic Farm
Data Points Parsing & CSG-based Clustering
CSG-based Grammar Tree Parsing
Grammar Graph Classifying
<Square>
<Parallelogram>

Fig. 1: In metalevel tracking, the trajectory data points from a target tracking radar are transformed into SCSG trees, which are subsequently classified into distinct target intent classes. The trajectories displayed are superimposed on the Google map for the Cornell University campus.

by SCSGs such as squares, equilateral triangles, etc., and by concatenating SCSG trajectories, one can construct more more complex trajectories.

Natural Language Processing (NLP) models and associated statistical signal processing algorithms have been used in trajectory analysis. Stochastic Context-Free Grammars and Reciprocal Process Models are utilized in [7], and have been extended to more complex metalevel tracking scenarios in [8], [9]. Syntactic tracking using GMTI measurements is studied in [19], [22]. However, stochastic context-free grammars, while more general than Hidden Markov Models (HMM), are more restrictive than SCSG. The potential of SCSG-based models and associated statistical signal processing for inference/classification for metalevel tracking remains largely unexplored.

Although Bayesian inference for SCSGs is NP-hard, the representation of grammar rules as graphs allows us to exploit the capabilities of Graph Neural Network (GNN) [6], [15] for efficient classification of SCSGs. By encoding syntactic structures as graphs, we enable GNN to effectively learn and identify patterns within SCSGs, enhancing trajectory analysis performance. Additionally, this approach benefits from the interpretability of grammar rules, leading to higher classification accuracy compared to directly classifying the texts generated by SCSGs using text-based methods [12], [21], [25].

Our key contributions are as follows:
-We utilize SCSG to model and parse object trajectories, capturing complex dependencies that are challenging for traditional methods.
- We classify trajectory intent using SCSG-derived grammar trees, providing a more accurate and interpretable framework compared to vision, vectorized-data or text based models.
-Our approach achieves higher accuracy, consistently outperforming deep learning models and highlighting the robustness of graph-based classification over text-based methods and other baselines.

II. Modeling Trajectory Intent with Geometric Shapes

In this section, we outline our approach for analyzing trajectory intent using geometric shape-based representations and stochastic grammar modeling. We begin by presenting a Bayesian trajectory framework that incorporates state-space models for target tracking. Next, we describe how basic geometric shapes—such as straight lines, loops, and zigzags—serve as building blocks for inferring the intent of complex trajectories with the help of stochastic grammars.

A. Target Dynamics and Observation Model

In classical target tracking radars [2], [3], the kinematic state of a target (position and velocity) at time k is represented by $\mathbf{x}_k = [p_k^1, \dot{p}_k^1, p_k^2, \dot{p}_k^2]^\top$ where p_k^i is the target's position in the i^{th} dimension at time k, and its observation is given by $\mathbf{y}_k \in \mathbb{R}^4$. The state specifies the target's position and velocity in a 2-dimensional space. The state evolves as

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{G}\mathbf{a}_k + \mathbf{w}_k, \tag{1}$$

where $\mathbf{F}$ and $\mathbf{G}$ matrices are defined in [18, Ch. 2.6]. The process noise is denoted as $\mathbf{w}_k \sim \mathcal{N}(0, Q)$, Q defined in [18, Ch. 2.6]. The acceleration $\mathbf{a}_k = [a_k^1, a_k^2]^\top$ ultimately determines the trajectory of the target, where a_k^i is the acceleration in the i^{th} dimension at time k. The observation at time k is

$$\mathbf{y}_k = \mathbf{x}_k + \mathbf{v}_k, \tag{2}$$

where $\mathbf{v}_k \sim \mathcal{N}(0, R)$ is measurement noise [18, Ch. 2.6].
Target Tracker. A Bayesian tracker computes the posterior

$$\Pi_{k+1} = \mathcal{B}(\Pi_k, \mathbf{y}_k). \tag{3}$$

Here $\mathcal{B}$ is the optimal filtering recursion or a sub-optimal approximation, such as a particle filter or IMM algorithm. (Since the maneuvers $\mathbf{a}_k$ are unknown and often modeled as a Markov chain, we have a non-linear filtering problem.) The estimated state at time $k+1$, denoted as $\hat{\mathbf{x}}_{k+1}$, is

$$\hat{\mathbf{x}}_{k+1} = \Phi(\Pi_{k+1}), \tag{4}$$

where $\Phi(\cdot)$ is the state estimation function that derives the most likely state from the updated posterior distribution.

The above abstraction depicts the state space model and target tracking algorithm. For simplicity, we assume a single target with with known clutter and perfect data association. In the remainder of this paper, the sequence $\{\hat{\mathbf{x}}_k\}$ will be utilized to infer the intent of the target.

B. Geometric Shape-Based Intent Modeling Using Stochastic Grammars

This section demonstrates how the intent of a target can be modeled by framing the estimated track sequence $\{\hat{\mathbf{x}}_k\}$ into geometric shapes. We focus on a higher level of abstraction, involving meta-level tracking. The key idea is to use a Stochastic Grammar from Formal Language Theory to fit geometric shapes as foundational elements for trajectory modeling to the sequence of track estimates $\{\hat{\mathbf{x}}_k\}$ generated above.

Stochastic Grammars: A stochastic grammar is defined as $G = (\mathcal{A}, \mathcal{E}, \Gamma, P_s, S_0)$, where $\mathcal{A}$ is the alphabet, $\mathcal{E}$ the set of non-terminal symbols, Γ the set of production rules, and S_0 the starting symbol and P_s is a probabilistic map defining the distribution over the production rules Γ. These grammars are stochastic because each non-terminal has multiple production rules, with the selection made randomly.

Generative Models for Trajectories. By a generative model for a trajectory, we mean a grammar that can exclusively generate these trajectories. This is typically verified using pumping lemmas [13]. Such shapes include:
- **Straight Line:** Represents direct, purposeful movement,
- **Rectangle:** Simulates a patrolling or scanning behavior,
- **Zigzag:** Suggests evasive or exploratory movement.

These shapes are combined in sequence to form composite trajectories. For example, a trajectory on a digital roadmap (google maps) can start with a straight line (approach), followed by a closed loop (survey), and end with another straight line (exit). Such combinations yield complex and realistic movement patterns. We use SCSG to model such complex trajectories.
Stochastic Regular Grammars (SRGs) are equivalent to Hidden Markov Models (HMMs) and serves a a generative model for a line trajectory.
Stochastic Context-Free Grammars (SCFGs) can be used to model branching or hierarchical trajectories, where decisions or splits in movement occur, such as navigating through a set of checkpoints with multiple paths.
Stochastic Context-Sensitive Grammars (SCSGs) can be used to model more complex trajectories that may involve dependencies between different parts of the trajectory, such as closed-loop or interacting trajectories with constraints on the relationships between different segments. Such long range dependencies are crucial for capturing real-world interactions and constraints, leading to more accurate intent prediction.

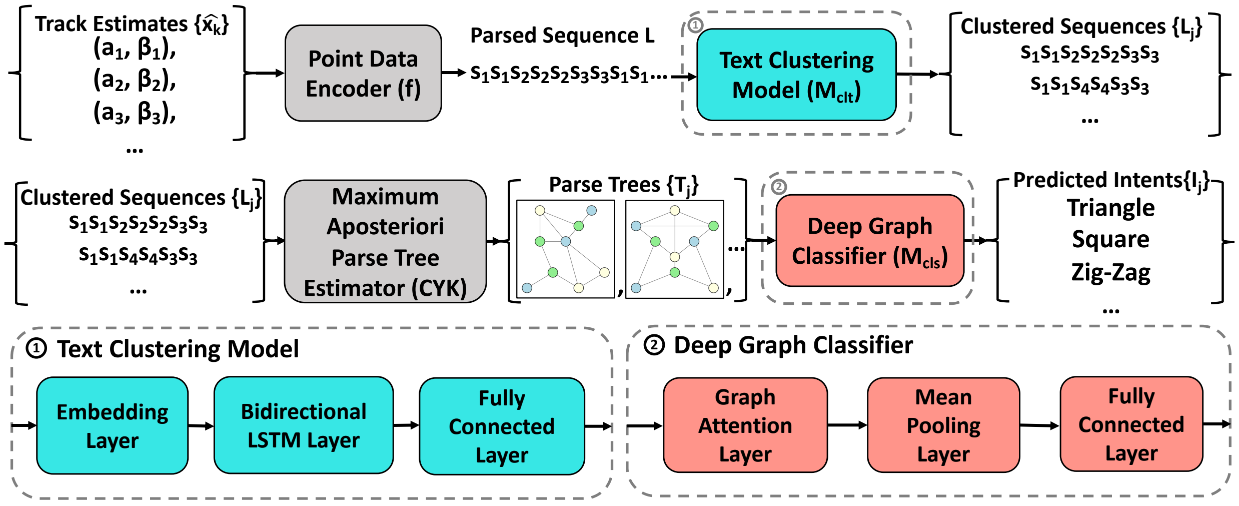


Fig. 2: Schematic of the meta-level tracking proposed in this paper includes the conversion of raw trajectory data into a string representation which is then segmented into sub-strings corresponding to distinct trajectory segments. These sub-strings are then parsed into grammar trees and classified into basic geometric shapes, each representing a unique movement intent.

Below we utilize SCSGs for trajectory modeling and analysis. Let $\mathcal{A} = \{s_1, s_2, s_3, s_4, ...\}$ to be a finite set of terminals, where each element within represents a unit acceleration of the target in a specific direction. Recall that the acceleration $\mathbf{a}_k$ defined in (1). We define $\mathcal{E} = \{S_1, S_2, S_3, S_4, ...\}$ to be a finite set of non-terminals, where each element within represents an intermediate state of the target. For further details on the production rules Γ for SCSG refer to [11]. Given an S_0, guided by stochastic production rules Γ and distribution over production rules P_s, a string $\{s_i\}$ of Stochastic Context Sensitive Language (SCSL) can be generated.

III. Parse-Tree based Deep Clustering and Classification of Trajectory

A naive approach to classify the trajectory of a target given the track sequence $\{\hat{\mathbf{x}}_k\}$ is to train a deep classifier. But this does not exploit the parse tree structure of a SCSG. In this section, we exploit the parse tree structure to classify target intent, as illustrated in Fig. 2. Our approach involves converting trajectory data into a structured textual representation, segmenting this text into sub-trajectories, reconstructing grammar trees, and classifying the intent of each sub-trajectory.

Trajectory Encoding: The estimated states $\{\hat{\mathbf{x}}_k\}$ are represented as a sequence of data points $\{(\alpha_i, \beta_i)\}_{i=1}^{k}$, where each (α_i, β_i) corresponds to an x, y coordinate. We convert this sequence into a textual representation, where each segment is encoded as a symbol $s_i = f(\alpha_i, \beta_i, \alpha_{i+1}, \beta_{i+1})$, reflecting the geometric displacement between consecutive points. The entire entire trajectory is represented as a string $L = s_1 s_2 \dots s_{k-1}$, modeled using SCSGs as introduced in Sec. 2. This string-based encoding allows for efficient parsing and clustering of the trajectory.

Deep Clustering Architecture on String: We then cluster L into sub-trajectories L_j using a text clustering model M_{clt}, which consists of an embedding layer, a bidirectional Long Short Term Memory (LSTM) layer, and a fully connected layer to produce sub-trajectory clusters. The embedding layer converts each character in the input sequence into a dense vector representation. The bidirectional LSTM layer captures both forward and backward dependencies in the sequence for better context understanding. Finally, the fully connected layer maps the LSTM output to cluster labels. The model is designed to minimize intra-cluster dissimilarity and is trained using cross-entropy loss, where the clustering is defined as $L_j = M_{\text{clt}}(L)$. The loss function is given by $\mathcal{L}_{\text{clt}} = -\sum_{j=1}^{m} \sum_{l=1}^{L} u_{jl} \log \hat{u}_{jl}$, where u_{jl} is the ground truth label, and $\hat{u}_{jl}$ is the predicted probability for segment L_j belonging to cluster l.

Grammar Tree Parsing: Each sub-trajectory L_j is used to reconstruct a grammar parsing tree T_j through the Cocke-Younger-Kasami (CYK) algorithm [14], which is an extension of the Maximum Aposteriori (MAP) based Viterbi algorithm, applying dynamic programming to parse Stochastic Context-Free Grammars (SCFGs). While the CYK algorithm is optimal for SCFGs, it is suboptimal for SCSGs, which involve more complex production rules. CYK is designed for SCFGs where rules permit binary splits and are independent of surrounding contexts, making it unsuitable for the length-increasing productions and context dependencies inherent in SCSGs. To address this, we limit the recursion depth of non-terminals in SCSG-based generation to manage the increased complexity effectively. Additionally, we extend the CYK parsing table to include contextual information, enabling it to track substring generation within specific contexts. This adaptation allows us to approximate the grammar parsing tree as $T_j = \text{CYK}(L_j, G)$, although the method remains suboptimal for SCSGs due to its inherent complexity.

Deep Classifier Architecture which Exploits Parse Tree: The tree structure T_j is then classified to determine the intent of the corresponding sub-trajectory using a graph classification model M_{cls}. The Graph Transformer is a model designed to extend the success of transformers in natural language processing to graph-structured data by leveraging attention mechanisms [21]. It begins by aggregating features of each node in the graph or tree structure T_j from its neighboring nodes, ensuring local information is incorporated. A transformer layer with multi-head attention [21] is then applied to enhance the model's expressive power, facilitating efficient message passing and updating node embeddings. To generate a graph-level representation, a global pooling mechanism combines the node embeddings, which is subsequently fed into a fully connected layer for classification. The model is trained with cross-entropy loss, where the loss function is defined as $\mathcal{L}_{\text{cls}} = -\sum_{j=1}^{m}\sum_{l=1}^{L} z_{jl}\log \hat{z}_{jl}$, where z_{jl} is the ground truth label, and $\hat{z}_{jl}$ is the predicted probability for tree T_j being assigned to intent label l. Finally, the intent label for each sub-trajectory is given by $I_j = M_{\text{cls}}(T_j)$. The sequence of intent labels for the overall trajectory is $\{I_1, I_2, \ldots, I_m\}$, where each I_j represents a distinct intent associated with the trajectory.

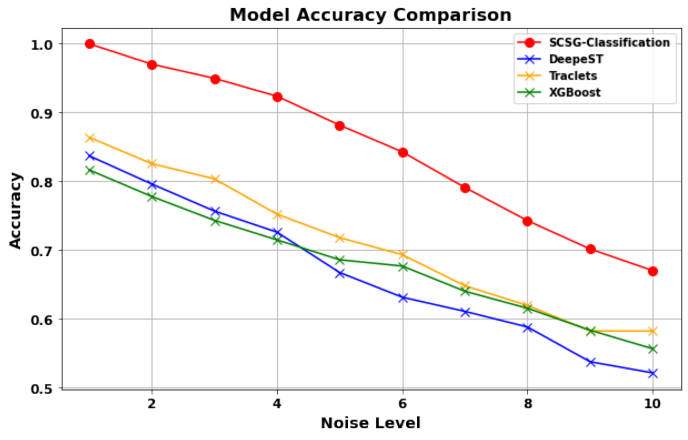
IV. Numerical Results

To evaluate the performance of our grammar-based graph classification model, we define the following key metrics and experimental conditions:

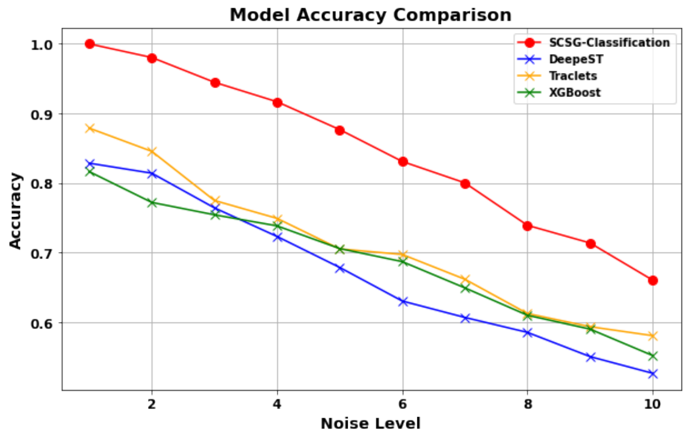
1) **Accuracy**: Accuracy measures the proportion of correct predictions made by the model over all predictions, reflecting its overall effectiveness in classifying trajectory intents under various conditions.
2) **Noise Level**: Noise level quantifies the degree of perturbation introduced by the grammar-based stochastic context-sensitive grammar (SCSG) generator. It is expressed on a scale from 1 to 10, where each level corresponds to a 5% probability of introducing deviations from the intended direction into each terminal of the grammar.
3) **Scenarios**: The experiments are conducted across three distinct scenarios, each requiring classification of intents with varying complexity as shown in Table I.

TABLE I: Scenario Parameters

Scenario and Description	Intents and Corresponding Shapes
Scenario 1: Binary Classification	- **Approaching:** Straight Line, Arc - **Patrolling:** Square, Rectangle
Scenario 2: Five-class Classification	- **Direct Moving:** Straight Line - **Turning:** L-Shape, V-Shape - **Patrolling:** Square or Rectangle - **Mapping:** Grid-Shape - **Random Walk:** Irregular Shape
Scenario 3: Ten-class Classification	- **Direct Moving:** Straight Line - **Winding:** Arc - **Turning:** L-Shape, V-Shape - **Diverting:** T-Shape - **Navigating:** Zigzag - **Patrolling:** Square, Rectangle - **Triangular Moving:** Triangle - **Spiral Moving:** Hexagon, Octagon - **Mapping:** Grid-Shape - **Random Walk:** Irregular Shape

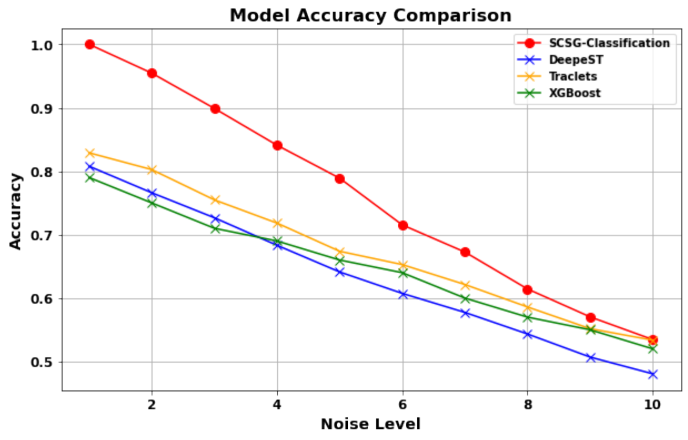


(a) Accuracy Comparison with other baselines at Scenario 1.



(b) Accuracy Comparison with other baselines at Scenario 2.

'



(c) Accuracy Comparison with other baselines at Scenario 3.

Fig. 3: Comparing accuracy across different noise levels, the graph-based model consistently exhibits higher accuracy than the vision-based model (TraClets), and the trajectory spatio-temporal-data-based model (DeepeST), and the vectorized-data-based model (XGBoost), indicating greater certainty in its predictions.

A. Comparison with other Methods

We compared our SCSG-based classification method with three baseline approaches, each employing distinct strategies for trajectory classification. DeepeST [5] is built around an LSTM network, explicitly designed to model the temporal dynamics of trajectories by embedding the target's spatial positions directly into the framework. It excels in integrating spatial and temporal features for intent classification, making it suitable for sequential trajectory tasks. TraClets [17] transforms trajectory data into visual representations, enabling the use of convolutional neural networks (CNNs) for classification. By leveraging image-based techniques, it identifies and analyzes spatial trajectory patterns with high precision. XGBoost [4] applies a gradient-boosted decision tree algorithm to classify trajectory state vector sequences. Its random forest-inspired strategy ensures an efficient balance between accuracy and computational performance when working with structured trajectory data.

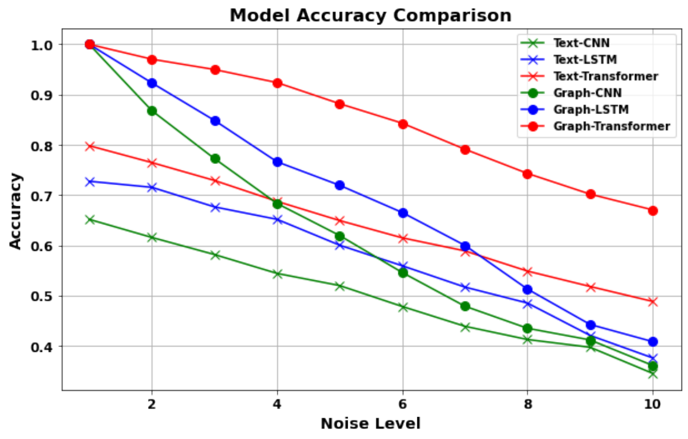
The comparisons presented are all based on the same clustering network, with our primary focus on evaluating the classification model. Comparisons over different scenarios and noise levels are displayed in Fig. 3a, Fig. 3b and Fig. 3c. Overall, the SCSG-Classification model exhibited superior performance in terms of accuracy across all noise levels, consistently outperforming the other models tested. While accuracy generally decreased for all models as noise levels increased, SCSG-Classification model maintained higher accuracy, indicating robustness against noise.

B. The Effectiveness of Grammar-tree Parsing

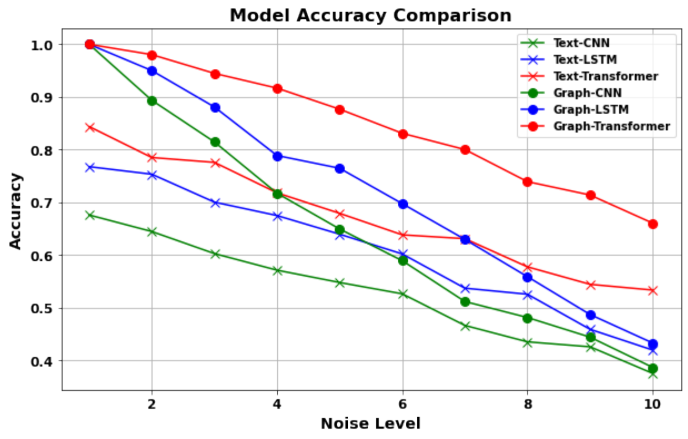
One may ask: why not directly classify the context-sensitive language instead of first obtaining the parse tree? To address this question, we conducted experiments to demonstrate the effectiveness of the Grammar-tree Parsing step. Beyond the transformer-based classifier, we implemented several alternative models for comparison. Notably, we developed a Graph Convolutional Network (Graph-CNN) model [6], [15], which utilizes convolutional layers to extract features from the graph structure. This approach aggregates node embeddings into graph-level representations through pooling, emphasizing spatial relationships within the graph and enabling direct classification based on these features.

Next, we developed a Graph-LSTM hybrid model, which extends the Graph-CNN by incorporating a Long Short-Term Memory (LSTM) network. In this approach, the Graph-CNN component extracts node-level embeddings, which are subsequently pooled into graph-level features. These features are then processed sequentially by the LSTM to capture temporal dependencies. This hybrid model enables a more comprehensive analysis, leveraging both the spatial relationships within the graph and the temporal patterns present in the data.

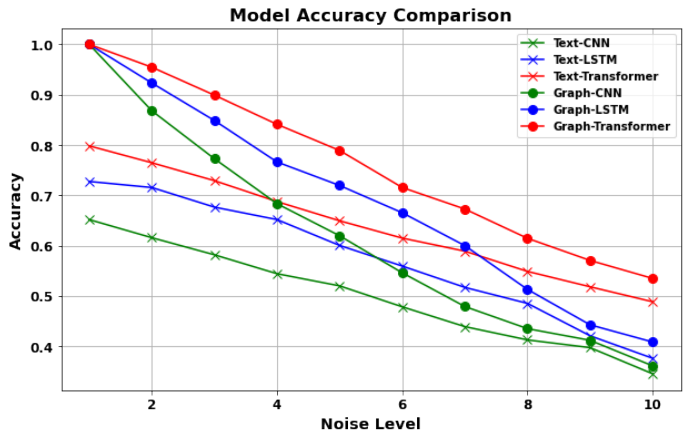
These models, along with the transformer-based classifier, were evaluated against their corresponding text classification methods to assess the impact of the Grammar-tree Parsing step compared to directly classifying raw input representations. The results, displayed in Fig. 4a, Fig. 4b, and Fig. 4c, highlight comparisons across various scenarios and noise levels. While accuracy generally declined for all models as noise levels increased, graph-based methods consistently outperformed their text-based counterparts, demonstrating greater robustness to noise.



(a) Graph-based Method vs. Text-based Method at Scenario 1.



(b) Graph-based Method vs. Text-based Method at Scenario 2.



(c) Graph-based Method vs. Text-based Method at Scenario 3.

Fig. 4: The graph-based methods (dot-lines) consistently achieve higher accuracy compared to the text-based methods ($\times$-lines), demonstrating superior performance in predictions.

V. Conclusion

In this paper, we presented a novel approach to trajectory intent analysis using Stochastic Context-Sensitive Grammars (SCSG) integrated with deep learning models that leverage grammar parse trees for radar-based metalevel tracking. By modeling trajectory patterns with SCSG and employing graph-based classification techniques, our method demonstrated significant improvements in accuracy over traditional spatio-temporal-data-based and vision-based trajectory classification approaches. Our results highlight the efficacy of grammar-aware classifiers in capturing complex spatio-syntactic dependencies, making them robust against noise and capable of providing meaningful insights into trajectory intent. Additionally, the use of graph neural networks and transformer-based architectures allowed for the effective utilization of the structured nature of parse trees, further enhancing classification performance.

Future work will explore the application of this framework in dynamic, multi-target scenarios and extend it to real-time implementations. Incorporating additional contextual information, such as environmental factors or interaction between targets, and optimizing parsing algorithms for SCSG inference could further improve the scalability and efficiency of the proposed method. This research opens new avenues for integrating formal language theory with deep learning for advanced trajectory analysis and intent prediction.

References

[1] S. Balari, A. Benítez-Burraco, M. Camps, G. Lorenzo, et al. Knots, language, and computation: A bizarre love triangle? replies to objections. *Biolinguistics*, 6(1):079–111, 2012.

[2] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[3] S. Blackman and R. Popoli. Design and analysis of modern tracking systems. *Artech House*, 1999.

[4] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[5] N. A. de Freitas, T. C. da Silva, J. F. de Macêdo, L. M. Junior, and M. Cordeiro. Using deep learning for trajectory classification. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART,*, pages 664–671. INSTICC, SciTePress, 2021.

[6] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[7] M. Fanaswala and V. Krishnamurthy. Detection of anomalous trajectory patterns in target tracking via stochastic context-free grammars and reciprocal process models. *IEEE Journal of Selected Topics in Signal Processing*, 7(1):76–90, 2012.

[8] M. Fanaswala and V. Krishnamurthy. Syntactic models for trajectory constrained track-before-detect. *IEEE Transactions on Signal Processing*, 62(23):6130–6142, 2014.

[9] M. Fanaswala and V. Krishnamurthy. Spatiotemporal trajectory models for metalevel target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 30(1):16–31, 2015.

[10] F. Ferrucci, G. Pacini, G. Satta, et al. Symbol-relation grammars: a formalism for graphical languages. *Information and Computation*, 131(1):1–46, 1994.

[11] K. S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[13] J. E. Hopcroft and J. D. Ullman. Introduction to automata theory, languages, and computation. In *Formal languages and their relation to automata*, pages 33–60. Springer, 1979.

[14] T. Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*, 1966.

[15] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[16] J. Kong, K. Zhang, and X. Zeng. Spatial graph grammars for graphical user interfaces. *ACM Transactions on Computer-Human Interaction*, 13(2):268–307, 2006.

[17] I. Kontopoulos, A. Makris, and K. Tserpes. Traclets: A trajectory representation and classification library. *SoftwareX*, 21:101306, 2023.

[18] V. Krishnamurthy. *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016.

[19] V. Krishnamurthy and S. Gao. Syntactic enhancement to vsimm for roadmap based anomalous trajectory detection: A natural language processing approach. *IEEE Transactions on Signal Processing*, 66(20):5212–5227, 2018.

[20] S. Oh, S. Russell, and S. Sastry. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 54(3):481–497, 2009.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[22] A. Wang, V. Krishnamurthy, and B. Balaji. Intent inference and syntactic tracking with gmti measurements. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4):2824–2843, 2011.

[23] X. Xie and R. Evans. Multiple target tracking using hidden markov models. In *IEEE International Conference on Radar*, pages 625–628. IEEE, 1990.

[24] D. Zhang, K. Zhang, and J. Cao. A context-sensitive graph grammar formalism for the specification of visual languages. *The Computer Journal*, 44(3):187–200, 2001.

[25] Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.