EXPLOITING UNCERTAINTY AWARE REASONING ABILITY OF LARGE LANGUAGE MODELS FOR CAD DRAWING COMPLETION

Anonymous authors
Paper under double-blind review

ABSTRACT

2D CAD drawings are essential documents for the design of industrial products. Their drawing process requires rigorous reliability assurance. However, traditional deep learning methods fail to provide convincing explanations when dealing with complex topologies. In this work, we propose a novel paradigm that leverages the powerful reasoning capabilities of LLM to drive cross-modal drawing completion, while enhancing the interpretability of LLM by designing a novel uncertainty-aware module. CAD drawings are encoded as text symbol sequences, and are subsequently integrated into instructional prompts to fine-tune LLM. Greedy Coordinate Gradient (GCG) is used to train adversarial suffixes, which are then used to execute adversarial attacks against LLM. Following this, we calibrate the LLM by utilizing the consistency metric of model outputs under perturbation. In order to quantify uncertainty, we devised a feature that incorporates a variety of metrics for evaluating intrinsic confidence to train the classifier. Our system achieves 93.6% accuracy in primitive classification and 70.9% accuracy in attribute prediction. In uncertainty analysis, our method first achieves efficient calibration, requiring only six perturbations and being applicable to both white-box and black-box models, subsequently achieving an AUC score of 0.919 in predicting model reliability. Experiments confirm our method's adaptability to mainstream models, and efficacy in addressing limited-sample learning scenarios and long-tail data distributions. Furthermore, we contribute a sketch dataset composed of instruction-prompt pairs, which includes 200,000 high-quality sketches, with the aim of contributing to research in this field.

1 INTRODUCTION

CAD drawings constitutes a cognitive process where designers translate abstract concepts into functional assemblies through structured geometric representations. As the basis for 3D modeling, sketches can be described by geometric elements (e.g., line, arc, circle, point) and constraint relationships (e.g., vertical, tangent, coincident) as shown in Fig. 1 (a). It may also be defined differently in other works.

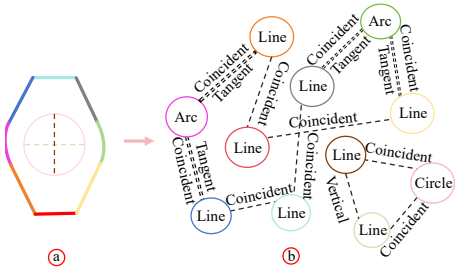


Figure 1: An example of CAD drawing and its graph form with primitives and constraints.

Manual drawing remains a labor-intensive process that requires multiple iterations across months. Prior researches have examined AI to accelerate the drawing, such as design tree method Camba et al. (2016), CNN Chen et al. (2024); Huang et al. (2016), and transformer structures Xu et al. (2022); Ganin et al. (2021); Seff et al. (2022); Karadeniz et al. (2024). These methods typically follow the paradigm: First, the CAD drawing's topology is mapped to a latent space via vectorized encoding, creating a representation with geometric features and constraint relations. Updating these vectors triggers the geometric constraint solver, which then maps the optimized results back to a geometry that re-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

spects engineering semantics. However, these approaches suffer from two limitations: (1) a lack of interpretability due to fuzzy associations between parameters and geometric components, and (2) generalization ability that is highly dependent on the scale and quality of training data, conflicting with the efficient sample learning requirements of real-world engineering.

In recent years, Large Language Model (LLM) have demonstrated significant effectiveness in cross-modal scenarios Wang et al. (2024c). For example, by establishing unified semantic representations to transfer the LLM's reasoning patterns to images Wang et al. (2024b); Liu et al. (2023); Qu et al. (2023) and audio Huang et al. (2024); Jiang et al. (2024), and by encoding graph structures as natural language Chai et al. (2023), LLM can solve graph-specific tasks. These provide confidence for our works and lead to the following considerations: 1) How to achieve an equivalent mapping between sketches and LLM, and transfer the LLM's reasoning capabilities to geometric logic? 2) Given the requirement for sample-efficient learning in CAD engineering, can the few-shot learning capabilities of LLM be effective? 3) To meet the high reliability requirements of CAD engineering, how to design a comprehensive approach to measure the reliability of model predictions.

SketchGraphs Seff et al. (2020) first introduced a large-scale parametric sketch dataset. Each sketch is represented as a geometric constraint graph composed of primitives and constraints (i.e., primitives are nodes, and constraints are edges), as shown in Figure 1 (b). We first reconstruct the geometric constraint graph and integrate geometric primitives and construction sequences. Then, these, along with the constraints, are encoded into a text sequence, and the LLM is fine-tuned through carefully designed instructional prompts. By applying the LoRa parameter-efficient fine-tuning Hu et al. (2022), only about 0.5% of the LLM's original weights were updated. In our experiments, using only 1,000 training samples, the LLM achieved an accuracy of 92.2% in predicting missing primitive types. Existing consistency-based uncertainty schemes suffer from efficiency limitations. For example, temperature Xiong et al. (2023), dropout Mo & Xin (2024), often require tedious hyperparameter tuning, Qian et al. (2024) requires laborious perturbation settings (10 perturbations are required). Therefore, we propose a new scheme. First, we use greedy gradient optimization Zou et al. (2023) to train adversarial suffixes, and by applying adversarial attacks to LLM, consistency metric are obtained. Then kernel density estimation is used to estimate the probability density distributions of the metric for positive and negative samples, and the information entropy calculated based on the density values allows the model to be calibrated. Experiments demonstrate that our scheme is efficient, requiring only an average of six perturbations, and is applicable to both white-box and black-box models.

Although confidence estimation methods based on analyzing logits Huang et al. (2023); Manakul et al. (2023); Brown et al. (1990) can provide some indication of confidence, they do not perfectly reflect cognitive certainty. Therefore, we propose a novel solution: designing a feature that incorporates a variety of metrics for evaluating intrinsic confidence to train the classifier, thereby quantifying uncertainty. Our method achieves a state-of-the-art baseline AUC score of 0.919 in predicting model reliability. Our contributions can be summarized as follows:

1. We propose a new paradigm for CAD drawing completion that leverages the reasoning capabilities of pre-trained language models.
2. We demonstrate the adaptability of this method to mainstream models, as well as its effectiveness in addressing limited-sample learning scenarios and long-tailed data distributions.
3. We introduce a novel uncertainty-aware module including a efficient calibration scheme based on GCG adversarial attacks, and a method for quantifying uncertainty which employs a classifier trained on the feature incorporating multiple metrics for evaluating intrinsic confidence.
4. We contribute a sketch dataset composed of instruction-prompt pairs, which includes 200,000 high-quality sketches.

2 Related Works

2.1 CAD Drawing Encoding

There are two major approaches to encoding CAD drawings. The first involves reformatting the primitive and constraint information into a vectorized format. Vitruvion Seff et al. (2022) proposes

vectorizing CAD drawings into sequences of primitives and constraints. These sequences are then processed through normalization, quantization, and tokenization to create algorithm-friendly formats. Willis et al. Willis et al. (2021) propose a sketch hypergraph representation where primitives are represented as a set of nodes, with edges connecting the nodes. However, this representation has the disadvantage of not encoding constraint information. The second approach leverages deep learning algorithms to encode CAD drawings into embeddings. SkexGen Xu et al. (2022) utilizes the power of transformers by building three encoding models that generate embeddings for topological, geometric, and extrusion information, which capture much richer details.

2.2 Fine-tuning Pre-trained Model

Once the CAD drawing is encoded into a vector embedding, the resulting vector can be fed into a large model to perform downstream tasks. Due to the complexity of CAD drawings, encoder-decoder transformers Wu et al. (2021); Xu et al. (2022) and GCNs Jones et al. (2023) have been popular choices. Alternatively, CAD drawings can be converted into images and processed by vision-based models such as TinyLLaVA, allowing them to follow instructions for modifying the CAD drawing Yuan et al. (2024) or to perform completion tasks Wang et al. (2024a).

Training large models, especially vision-language models, typically requires substantial datasets to achieve high-quality results. Our work explores the feasibility of using a purely pre-trained language model as the backbone, leveraging its inherent logical reasoning capabilities. This approach potentially reduces the reliance on large-scale data, offering a more data-efficient alternative to current methods.

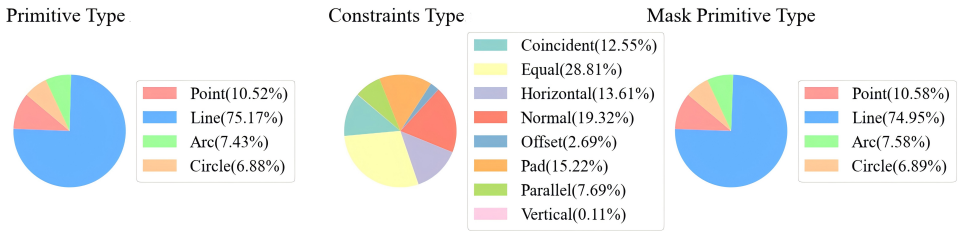


Figure 2: Distribution of primitive types, constraint types, and mask primitive types in our dataset.

2.3 Uncertainty Estimation

Hallucination is one of the critical challenges in GenAI systems, causing them to produce erroneous or false content in response to prompts. One method to detect hallucinations is to measure the model's confidence or uncertainty level. Higher uncertainty indicates lower confidence, implying a greater likelihood of a false output. Analyzing token probabilities from the GenAI output is a popular approach to evaluating this uncertainty. Huang et al. Huang et al. (2023) suggest that the minimum, maximum, and average token log probabilities correlate with the uncertainty of the model's responses. Beyond token probabilities statistics, various forms of entropy calculated from these probabilities Kadavath et al. (2022); Kuhn et al. (2023); Duan et al. (2024) serve as additional indicators of output uncertainty. Another way to evaluate uncertainty involves consistency measures, where small perturbations are introduced, such as changes in temperature Xiong et al. (2023), dropout Mo & Xin (2024), or random token alterations Qian et al. (2024), to inject noise into the model's generation process. The resulting deviation in the model's responses serves as a measure of its underlying confidence. In this paper, we propose a novel method based on greedy gradient optimization. It avoids laborious hyperparameter tuning and inefficient random perturbations. Moreover, it's applicable to both white-box and black-box models. Thereby providing a stronger basis for evaluating uncertainty.

3 Method

3.1 Datasets

From a subset of the SketchGraphs dataset Seff et al. (2020), which originally contained 15 million CAD drawings, we constructed a dataset of 200,000 high-quality CAD drawings. We followed

Seff et al. (2022) to deduplicate, normalize, and quantize the original dataset, then selected 200,000 relatively complex sketches from the processed dataset and ensured that each selected drawing contained at least 10 primitives. To facilitate model training, we randomly masked 10% of the primitives and used the types and attributes of the masked primitives as labels. Our dataset was evenly divided into 50% for training and 50% for testing. The distribution of primitives and constraints is shown in Figure 2. This dataset also has a long tail problem. Among them, Line accounts for 75.17% and Equal accounts for 28.81%, which is much higher than the smallest Vertical.

3.2 Automated Drawing Completion

3.2.1 Encoding Graph Structure

The geometric constraint graph is a graph structure with hierarchical characteristics. Its node set is composed of primitive nodes and sub-primitive nodes (endpoint) and the edge set contains three types of constraint relationships: geometric relationship constraint, numerical parameter constraint, and hyperedge constraint involving more than two reference nodes. The definition of the geometric constraint graph is described as follows: $G = (V, E)$, where $V = \{v = (t_v, c_v) \mid t_v \in \Gamma_p \cup \Gamma_s\}, E = \{e = (t_e, C_e, a_e) \mid t_e \in \Gamma_e, C_e \subseteq V\}$ Where: Γ_p, Γ_s, Γ_e represent the type space of geometric primitives, sub-primitives and constraints respectively. c_v is the coordinate parameter of the primitive, and a_e is the numerical parameter of the constraint. C_e represents the connected node set. Geometric primitives are forced to associate sub-primitives, which increases the complexity of the graph and raises the risk of excessively long encoded contexts. Given that sub-primitives are derived from geometric primitives, we prune sub-primitives and their constraints, resulting in a reconstructed geometric constraint graph.

$$G^* = (V_p, E^*) \qquad E^* = \{e \in E \,|\, \forall v \in C_e, v \in V_p\} \tag{1}$$

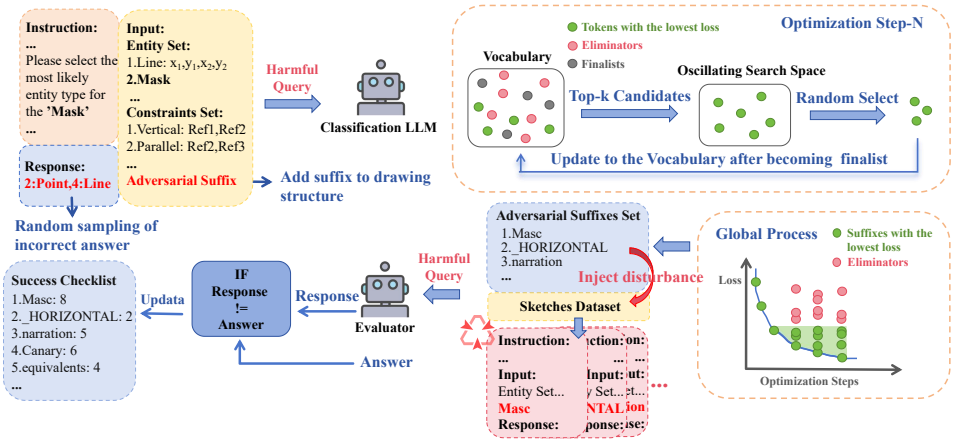


Figure 3: Flowchart for obtaining adversarial suffixes.

Geometric primitives are integrated with constructive sequences and encoded into a text sequence along with constraints. Following the Alpaca instruction tuning template Taori et al. (2023), text sequences and well-designed prompts (Table 5) are incorporated to obtain fine-tuning instructions.

3.2.2 Fine-tuning and Inference

Llama 3.1 8B Grattafiori et al. (2024) is used as the base model. The primitives in the sketch are randomly masked and their types and attributes serve as labels. In order to simplify the LLM's inference, we decouple this task into two sub-tasks: classification prediction and coordinate reasoning, and build independent pre-trained models for each sub-task. These models are trained on ground truth data to learn to infer the type and coordinates of the masked primitives from known primitives and constraint. The task-specific objective function is described as:

$$\theta^* = \arg\min_{\theta'} \sum \mathcal{L}(\text{LLM}(V_p \backslash V_m, E^*, \mathcal{T}, I; \theta'); \text{Response}) \tag{2}$$

Where V_m represents the set of mask primitives, $\mathcal{T}$ represents the primitive constructive sequences, and I represents the instruction. The goal is to find the parameter θ^* with the minimum loss on the dataset. The inference stage is a two-stage pipeline: first, the primitive classification model classifies the masked primitives, embedding the classification results into the encoded drawing structure. The attribute inference model then performs attribute inference on these primitives.

3.3 Uncertainty Analysis

Greedy Coordinate Gradient (GCG) is an attack method against LLMs, which aims to induce them to output harmful content by generating specific suffixes or prompts. To evaluate the fine-tuned model's geometric knowledge, we introduce an uncertainty-aware module based on GCG adversarial attacks, centered on injecting adversarial suffix perturbations, a calibration scheme, and a method for estimating confidence scores.

3.3.1 Training Adversarial Suffixes

We randomly select n samples from the sketch dataset and construct an adversarial suffix training set by changing the labels to incorrect answers, and the classification LLM is used as the base model. GCG directly adds harmful suffixes to the original query. In contrast, we focus on the drawing structure, adding suffixes to it in order to induce model instability by confusing the model about the drawing. Through gradient calculation, we select the top-k tokens with the smallest loss as candidates:

$$\text{Candidates} = \arg\min_{v \in \mathcal{V}}^{(k)} e_v \cdot \left(\frac{g_i}{\|g_i\|}\right) \tag{3}$$

Randomly select suffixes of this step from the candidates to avoid losing potential suffixes. This process will iterate N times. The loss function can be described as follows:

$$\mathcal{L}(x_{n+1:n+h}) = -\log \prod_{i=1}^{h} p(x_{n+i} \mid x_{1:n+i-1}) \tag{4}$$

Where h represents the length of the suffix. The generated suffixes are screened by the evaluator, and the number of "successful jailbreaks" is used as the criterion to maintain the suffix checklist. The top ranked suffixes will be extracted, resulting in an adversarial suffix set. The flowchart can be seen in Figure 3.

3.3.2 Uncertainty Calibration

For sample x, the LLM is perturbed by adding the adversarial suffix. The cumulative probability p of the output is then collected. This process is repeated for each adversarial suffix, resulting in a probability distribution $P = \{p_1, p_2, ...p_k\}$. The standard deviation reflects the degree of dispersion of a distribution. Separate positive and negative samples and calculate the negative log standard deviation separately, which can be described as:

$$-\log \sigma_x = -\log\left(\sqrt{\frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{P})} + \varepsilon\right) \tag{5}$$

The smoothing factor ε alleviates the data distribution deviation caused by negative logarithmic transformation. Since $-\log \sigma$ does not satisfy the normal distribution, we suggest using kernel density estimation (KDE) to estimate the probability density (PDF). Each sample x_i is considered as an independent variable distributed identically. PDF can be expressed as:

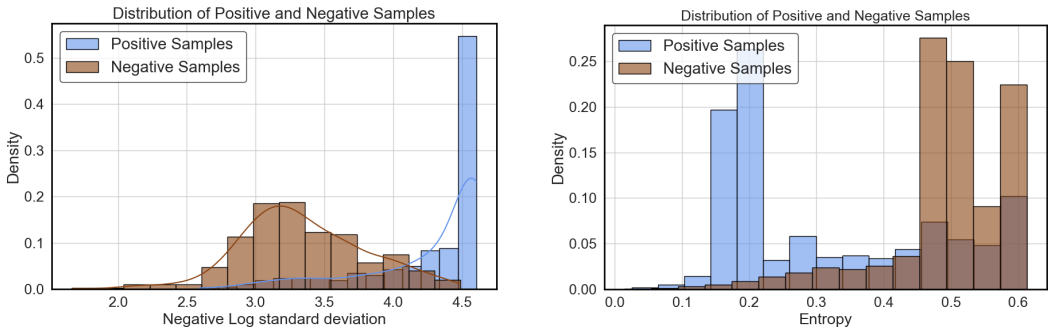
$$f(x) = \frac{1}{nh}\sum\nolimits_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{6}$$

where n denotes the data size, K is the Gaussian kernel, and the bandwidth h is determined using Scott's rule. Construct PDFs f_1 and f_2 for positive and negative samples. The histogram and probability density curve can be seen in Figure 4a. Given the referenced PDFs, a test unit t can obtain positive and negative class likelihood pairs $(p1, p2)$ via the negative log standard deviation. Information entropy Shannon (1948) reflects the uncertainty of random variables. We hypothesize

that calculating the entropy of $p1$ and $p2$ can reflect the LLM's uncertainty on the test set. The process of calculating the entropy of t is described as follows:

$$H(t) = -\sum_{i=1}^{2} p_i \, log \, p_i = -\sum_{i=1}^{2} f_i(t) \, log \, f_i(t) \tag{7}$$

The distribution of entropy is shown in Figure 4b. For samples with correct inference, the output of LLM has low deviation, resulting in lower entropy. Conversely, for incorrect samples, the LLM exhibits higher uncertainty.



(a) Histogram and probability density curve.

(b) Entropy distribution on the test set.

Figure 4: Distributions used for uncertainty calibration.

3.3.3 Confidence Estimation

One challenge in evaluating the confidence of LLM is the lack of objective anchors for reference benchmarks. To address this, we propose a novel solution: designing a feature vector that integrates intrinsic confidence estimates from multiple metrics to train a logistic regression classifier, and use the correctness of the classification as labels. The composition of the feature encompasses four mainstream methods: probability uncertainty, entropy uncertainty, consistency uncertainty (standard deviation based on GCG), and Perplexity. The definition of Perplexity is as follows:

$$\text{PPL} = exp(-\frac{1}{N}\sum_{i=1}^{N} log \, P(x_i \, | \, x_1, ...x_{i-1})) \tag{8}$$

This fusion mechanism effectively addresses the problem of missing dimensions in evaluating model confidence in prior works.

4 Results

4.1 Experiments Setup

Our experiments were conducted on a server equipped with an Nvidia 4090 GPU and an Intel(R) Xeon(R) Silver 4310 CPU @ 2.10GHz. We trained our model using half-precision (bf16) and a learning rate of 0.0003. For datasets with more than 10,000 samples, we used 1 epoch in the training, while for smaller datasets, performance peaked at around 20 epochs. To reduce resource requirements, we applied LoRA Hu et al. (2022) fine-tuning with a lora_rank of 32, lora_alpha of 64, and lora_dropout of 0.05.

Missing Primitive Type Identification

Table 1: Comparison of missing primitive type identification from state-of-the-art algorithms.

Model	ACC (Primitives)	Macro-F1	Weighted-F1
Ours	**0.9361**	**0.8698**	**0.9344**
RGCN	0.8845	0.7545	0.8779
RGAT	0.7511	0.2565	0.6572
HGCN	0.7365	0.4118	0.7707
HAN	0.7556	0.2152	0.6503
HGT	0.8926	0.7685	0.8863
GraphSAGE	0.9030	0.8136	0.9029
GCNII	0.7524	0.5616	0.822
TAM	0.5966	0.3476	0.6209

To verify the effectiveness of missing primitive type identification, we evaluate the accuracy of reasoning in units of primitives (ACC Primitives). Since our task belongs to multi-classification, we also use Macro-F1 and Weighted-F1 as evaluation metrics. The calculation of all evaluation metrics is summarized in Appendix A.

Table 2: Primitive-wise classification accuracy.

Model	Point 10.52%	Line 75.17%	Arc 7.43%	Circle 6.88%
Ours	0.8818	**0.9788**	**0.7807**	**0.7301**
RGCN	0.7664	0.9597	0.7402	0.3929
RGAT	0	0.9899	0.0969	0.006
HGCN	0.2591	0.9085	0.2942	0.0813
HAN	0	1	0	0
HGT	0.816	0.9627	0.7407	0.4024
GraphSAGE	**0.8958**	0.9446	0.7492	0.6223
GCNII	0.3128	0.9283	0.2073	0.1926
TAM	0.2824	0.2824	0.2619	0.2094

The sketch primitive classification has not been emphasized in prior works, so we propose a comparative approach by converting sketches into graph structures and training them using GNNs. We used eight approaches (RGCN Schlichtkrull et al. (2018), RGAT Wang et al. (2020), HGCN Chami et al. (2019), HAN Wang et al. (2019), HGT Hu et al. (2020), GraphSAGE Hamilton et al. (2017), GCNII Chen et al. (2020) and TAM Song et al. (2022)). The evaluation results are presented in Table 1. Leveraging the capabilities of LLMs, our system outperforms the strongest baseline, by 3.31% in ACC Primitives, 5.62% in Macro-F1, and 3.15% in Weighted-F1. When examining prediction accuracy by primitive type (Table 2), we found that the RGAT and HAN methods were severely affected by overfitting. In contrast, our approach effectively addresses imbalanced class distribution. The primitive distribution is highly skewed toward lines, so the per-class breakdown is useful for separating models that benefit from the dominant category from models that retain accuracy on minority primitives. Our method maintains stronger performance on arcs and circles while preserving high line accuracy, which supports the robustness of the text-based formulation under long-tailed data.

4.2 Attribute Generation

Table 3: Attribute generation across model sizes.

Model	ACC (Primitives)	ACC (Sketch)
Llama 3.1 - 8B	**0.7091**	**0.5624**
phi 3.5 mini - 3.8B	0.6556	0.5299
Llama 3.2 - 3B	0.6612	0.532
Gemma 2 - 2B	0.6566	0.5345
Llama 3.2 - 1B	0.6005	0.4642

The second-stage CAD drawing completion process takes the predicted missing primitive labels and incorporates them into the instruction prompt to generate the attributes of the missing primitives. Compared with type identification, this stage requires the model to infer continuous geometric parameters from the surrounding symbolic context, including endpoint coordinates, circle centers, radii, and arc control points. We evaluate mainstream models with different parameter sizes, and use accuracy on primitives and accuracy on sketch as metrics. Their calculation is provided in Appendix A. The Llama 3.1 8b model achieves a primitive accuracy of 0.7091 and a sketch accuracy of 0.5624. The model-size comparison in Table 3 shows that the performance of the attribute generation task is correlated with the model size, indicating the potential reasoning ability of the LLM. Smaller models can still produce valid output formats, but they are more likely to fail when multiple constraints must be jointly satisfied. This suggests that attribute generation depends not only on local token prediction, but also on maintaining geometric consistency across the entire sketch.

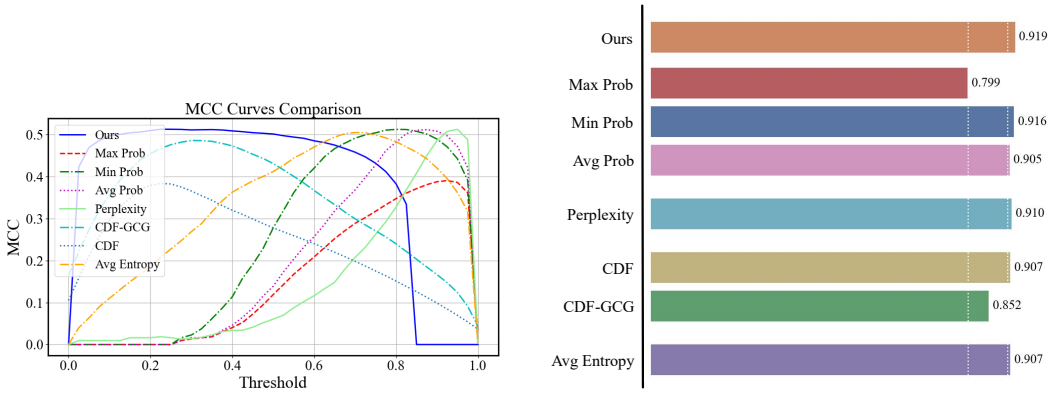
4.3 Uncertainty Estimation

The experimental setup is as follows: we extract 1,000 samples (including 600 positive and 400 negative) from the training set for suffix generation and classifier training. Then we generate 1,000 suffix templates; following screening by the evaluator, the top-10 are retained as the adversarial suffix set. The resulting feature weights (Max Prob, Min Prob, Avg Prob, GCG Std, Perplexity, Entropy) are $[-0.51, 0.86, 0.34, -0.37, -1.11, -2.06]$. These weights provide insight into the relative influence of each feature on the prediction, with the assumption that the absolute value of the feature's weight indicates the lever of its impact. We observed that Min Prob and Avg Prob exhibit positive influence. On the other hand, Entropy, Perplexity, and GCG Std displayed negative correlation. An exception is the Max Prob, for which the negative weight is counterintuitive. This is attributed to the tendency of LLM to consistently assign high confidence to the prediction of some primitives when

inferring sketches with multiple missing primitives. Consequently, we ultimately retained the Min Prob, GCG Std, Perplexity, and Entropy features.

In order to validate our method, we compare the dominant schemes for evaluating white-box models. Considering the imbalance of our data, we choose Matthews Correlation Coefficient (MCC) as metric, setting the threshold in steps of 0.05 and calculate the MCC score under each threshold. The MCC definition is included in Appendix A. The evaluation result is shown in Fig. 5a. In contrast to other methods, our approach consistently maintains high performance over a wide threshold range, exhibiting smooth, non-oscillating curves. This suggests that integrating multiple metrics effectively mitigates sensitivity to specific samples. The precipitous drop in MCC to zero for high thresholds (¿0.83) signifies a conservative state.

We further report Area Under Curve (AUC), whose calculation is summarized in Appendix A. As shown in Fig. 5b, our method surpassed the strongest baseline and achieved an AUC score of 0.919. Furthermore, the fact that six out of eight metrics achieved an AUC $\geq$ 0.9 demonstrates the fine-tuned LLM's acquired capability for anomaly detection during prediction, allowing it to prevent high-risk misclassifications.



(a) MCC curves. (b) AUC comparison.

Figure 5: Evaluation of uncertainty measures on our CAD drawing completion model. CDF-GCG represents replacing random perturbations with adding adversarial suffix perturbations.

5 Additional Analyses

We further evaluate the influence of training-set size and augmentation in Table 4. This experiment is moved from the appendix to the main text because it directly supports the sample-efficiency claim of our CAD-to-language formulation. The model already reaches 0.9202 primitive accuracy and 0.9176 weighted F1 with only 1,000 samples, indicating that fine-tuning can transfer useful geometric reasoning from limited task-specific data.

Table 4: Impact of training sample size.

# Samples	ACC (Primitives)	Macro-F1	Weighted-F1
600,000	0.9402	0.8537	0.9363
100,000	0.9361	0.8699	0.9344
50,000	0.9338	0.8653	0.9323
10,000	0.9303	0.8431	0.9294
8,000	0.9235	0.4838	0.9228
5,000	0.9205	0.4803	0.9195
4,000	0.9204	0.4801	0.9194
3,000	0.9204	0.4801	0.9194
2,000	0.9202	0.4732	0.918
1,000	0.9202	0.4719	0.9176

Performance generally increases as the training set grows, but the gain from simple order-based augmentation is limited: expanding the 100,000-sample set to 600,000 samples only marginally improves primitive accuracy and weighted F1. The lower Macro-F1 under small data also reflects the continuing difficulty of minority primitive categories. Additional deployment and training-configuration studies remain in Appendix B, where Table 7 compares precision and inference frameworks and Table 6 reports the training schemes for attribute generation.

5.1 Impact on Adversarial Attack Iterations

To explore the perturbation efficiency of our method, we analyze the impact of different attack iterations on model prediction confidence. We group adversarial suffixes based on the number of attack iterations. Since each group contains different combinations of suffixes, we use box plots to depict the AUC scores for different combinations (Figure 6). As the number of attack iterations increases, the distribution exhibits an upward trend, and the overall mean gradually increases. When the number of attack iterations reaches 6, the median and quartiles of the AUC nearly reach their peak, and then fluctuate around the peak as the number of attack iterations further increases. This trend indicates that a small number of optimized perturbation steps is sufficient to expose unstable predictions, while excessive iterations provide diminishing returns. The saturation after six iterations also suggests that the suffix search quickly reaches the most informative perturbation directions for this task. Therefore, the selected adversarial suffixes offer a practical balance between calibration strength and computational cost.

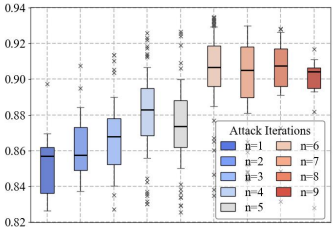


Figure 6: AUC score of different suffix combinations under each attack iteration.

6 Failure Mode Analysis

Three representative samples are illustrated in Fig. 7. In (a), the missing primitive types and their attributes are correctly identified. In (b), although the missing primitive type is correctly determined by the identification model, an error in the second attribute-generation model causes the rendered CAD drawing to become significantly distorted. In (c), the missing primitive type is identified incorrectly, resulting in a line being added instead of a point. This incorrect classification leads to the generation of incorrect attributes. These cases indicate that remaining errors mainly arise from stage-to-stage propagation: attribute errors can distort geometry even when the primitive type is correct, while type errors directly change the rendered topology. The examples also show that visual correctness is sensitive to small coordinate deviations, because a locally wrong attribute may change the global constraint layout after rendering. In contrast, primitive-type errors are more structural and are usually easier to identify from the final topology. These observations motivate future checks that jointly validate primitive type, attribute range, and rendered geometry before accepting a completion.

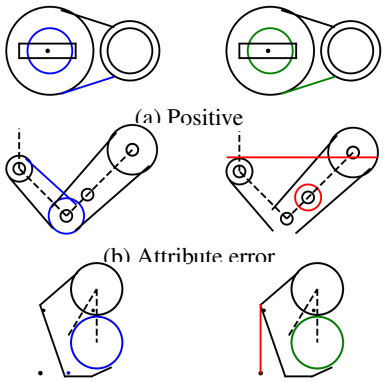


Figure 7: Failure Analysis on three samples.

7 Conclusion

In this paper, we first propose a new paradigm that leverages the logical reasoning and causal analysis capabilities of a pure language model to drive cross-modal drawing completion, effectively addressing the issues of limited interpretability and reliance on dataset found in previous works. Second, we introduce a novel uncertainty-aware module, which includes a calibration scheme based on greedy gradient optimization. This scheme avoids laborious hyperparameter tuning and inefficient random perturbations, and is applicable to both white-box and black-box models. This module also includes a method for quantifying confidence by designing a feature that incorporates a variety of metrics for evaluating intrinsic confidence to train the classifier, thereby addressing the problem of missing evaluation dimensions in traditional methods. Furthermore, we contribute a sketch dataset composed of instruction-prompt pairs, which includes 200,000 high-quality sketches, with the aim of contributing to research in this field.

AI USE STATEMENT

Generative AI tools may have been used for language polishing and LaTeX formatting assistance. All technical claims, experiments, citations, and AI-assisted edits should be reviewed and verified by the authors, who take responsibility for the final submitted content.

ETHICS STATEMENT

This work uses publicly available CAD sketch data and does not involve human subjects. Potential risks include unreliable generated CAD completions in safety-critical design contexts; therefore, predictions should be validated by domain experts and downstream CAD constraint checks before practical deployment.

REPRODUCIBILITY STATEMENT

We describe the dataset construction, model fine-tuning, uncertainty calibration, evaluation metrics, and ablation settings in the main paper. The instruction-prompt sketch dataset and implementation details are intended to support reproduction of the reported experiments.

REFERENCES

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 1990.

Jorge D Camba, Manuel Contero, and Pedro Company. Parametric cad modeling: An analysis of strategies for design reusability. *Computer-aided design*, pp. 18–31, 2016.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. Graphllm: Boosting graph reasoning ability of large language model. *arXiv:2310.05845*, 2023.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 4869–4880, 2019.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735, 2020.

Wu Chen, Hehe Fan, Qiuping Jiang, Chao Huang, and Yi Yang. Progressive point cloud denoising with cross-stage cross-coder adaptive edge graph convolution network. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 6578–6587, 2024.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 5050–5063, 2024.

Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. In *Advances in Neural Information Processing Systems*, pp. 5885–5897, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *ArXiv*, 2024.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1025–1035, 2017.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pp. 2704–2710, 2020.

Haibin Huang, Evangelos Kalogerakis, Ersin Yumer, and Radomir Mech. Shape synthesis from sketches via procedural models and convolutional networks. *IEEE transactions on visualization and computer graphics*, pp. 2003–2013, 2016.

Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. Audiogpt: Understanding and generating speech, music, sound, and talking head. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 23802–23804, 2024.

Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv:2307.10236*, 2023.

Yiyang Jiang, Wengyu Zhang, Xulu Zhang, Xiao-Yong Wei, Chang Wen Chen, and Qing Li. Prior knowledge integration via llm encoding and pseudo event regulation for video moment retrieval. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024.

Benjamin T Jones, Michael Hu, Milin Kodnongbua, Vladimir G Kim, and Adriana Schulz. Self-supervised representation learning for cad. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21327–21336, 2023.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv:2207.05221*, 2022.

Ahmet Serdar Karadeniz, Dimitrios Mallis, Nesryne Mejri, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Picasso: A feed-forward framework for parametric inference of cad sketches via rendering self-supervision. *arXiv:2407.13394*, 2024.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv:2302.09664*, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, pp. 34892–34916, 2023.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv:2303.08896*, 2023.

Shentong Mo and Miao Xin. Tree of uncertain thoughts reasoning for large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 12742–12746, 2024.

Zhenyu Qian, Yiming Qian, Yuting Song, Fei Gao, Hai Jin, Chen Yu, and Xia Xie. Harnessing the power of large language model for uncertainty aware graph processing. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 8035–8049, 2024.

Leigang Qu, Shengqiong Wu, Hao Fei, Liqiang Nie, and Tat-Seng Chua. Layoutllm-t2i: Eliciting layout guidance from llm for text-to-image generation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 643–654, 2023.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018*, pp. 593–607, 2018.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in cad. In *ICML Workshop on Object-Oriented Learning*, 2020.

Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric CAD sketches. In *International Conference on Learning Representations*, 2022.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, pp. 379–423, 1948.

Jaeyun Song, Joonhyung Park, and Eunho Yang. Tam: Topology-aware margin loss for class-imbalanced node classification. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 20369–20383, 2022.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3229–3238, 2020.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pp. 2022–2032, 2019.

Xiaogang Wang, Liang Wang, Hongyu Wu, Guoqiang Xiao, and Kai Xu. Parametric primitive analysis of cad sketches with vision transformer. *IEEE Transactions on Industrial Informatics*, pp. 12041–12050, 2024a.

Yabing Wang, Le Wang, Qiang Zhou, Zhibin Wang, Hao Li, Gang Hua, and Wei Tang. Multimodal llm enhanced cross-lingual cross-modal retrieval. 2024b.

Yabing Wang, Le Wang, Qiang Zhou, Zhibin Wang, Hao Li, Gang Hua, and Wei Tang. Multimodal llm enhanced cross-lingual cross-modal retrieval. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 8296–8305, 2024c.

Karl DD Willis, Pradeep Kumar Jayaraman, Joseph G Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2105–2114, 2021.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6772–6782, 2021.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv:2306.13063*, 2023.

Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In *The International Conference on Machine Learning*, pp. 24698–24724, 2022.

Zhe Yuan, Jianqi Shi, and Yanhong Huang. Openecad: An efficient visual language model for editable 3d-cad design. *Computers & Graphics*, pp. 104048, 2024.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv:2307.15043*, 2023.

A Evaluation Metrics

This section summarizes the calculation of the metrics used in the main experiments. For missing primitive type identification, primitive accuracy is computed as

$$Primitives\ Accuracy = \frac{N_p}{N_m}, \tag{9}$$

where N_p represents the number of correctly classified primitives, and N_m represents the total number of masked primitives.

For multi-class classification, we report F1, Macro-F1, and Weighted-F1:

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \quad F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \tag{10}$$

$$F1_{Macro} = \frac{1}{n} \sum_{i=1}^{n} F1_i, \quad F1_{Weighted} = \sum_{i=1}^{n} w_i \cdot F1_i. \tag{11}$$

Here, n denotes the number of classes, $F1_i$ is the F1 score of class i, and w_i is the class weight used in the weighted average.

For attribute generation, sketch accuracy is defined as

$$Sketch\ Accuracy = \frac{N_s}{N}, \tag{12}$$

where N_s is the number of sketches whose attributes are correctly inferred, and N is the total number of evaluated sketches.

For uncertainty estimation on imbalanced data, we use Matthews Correlation Coefficient:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{13}$$

In this equation, TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

We also use Area Under Curve (AUC) to evaluate performance across classification thresholds:

$$AUC = \int_0^1 TPR(FPR)\, dFPR, \quad TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}. \tag{14}$$

B Additional Tables

This appendix collects the supporting experiments and implementation details referenced in the main text. The tables are kept as compact inline blocks so that related results can share the same appendix pages without forcing a page break after every experiment.

Prompt design. Table 5 gives the two instruction templates used by the pipeline. The classification prompt restricts the prediction target to the four primitive categories used in SketchGraphs, while the attribute prompt fixes the response schema for each primitive type. Keeping these templates explicit is important because the output format determines whether the predicted attributes can be parsed by the CAD reconstruction step. The two prompts are intentionally concise: all geometric information is provided by the encoded sketch context, while the instruction only defines the expected task and output space. This design reduces ambiguity during fine-tuning and makes the same sketch representation usable for both classification and attribute inference.

Table 5: Instruction prompts for the two-stage pipeline.

 # Classification Prediction: The following is a 2D CAD sketch composed of entities and constraints. Please select the most likely entity type for the 'Mask' within the Entity Set. 1.Point 2.Line 3.Circle 4.Arc
 # Attribute inference: The following is a 2D CAD sketch composed of entities and constraints. Please give the coordinates of entity_{ } in the following format: Point:[x,y]; Line:[x1,y1,x2,y2]; Circle:[x of the center,y of the center,radius]; Arc:[x1,y1,x of the center,y of the center,x2,y2]

Attribute generation settings. Table 6 summarizes the fine-tuning objective used by the second-stage attribute generator. The model-size comparison is reported in the main text, while this appendix table focuses on whether mixing classification and attribute objectives improves the coordinate synthesis stage. We separate this comparison from the main attribute-generation table because it studies the training objective rather than the backbone capacity. In practice, the classification task and the attribute-generation task require different output distributions, and forcing them into a shared objective may weaken the coordinate-format discipline needed by the second stage.

Table 6: Fine-tuning schemes for attribute generation.

Configuration	ACC (Primitives)	ACC (Sketch)
Attribute generation only	**0.7091**	**0.5624**
Classification + attribute generation	0.6499	0.4805
Continual learning	0.7052	0.5576

The fine-tuning comparison shows that a focused attribute-generation objective works better than multi-task or continual training. This suggests that the second stage benefits from a clean objective centered on precise coordinate synthesis after the primitive type has already been provided, so the simpler single-task setting is used in the main pipeline. This result also supports the modular design of our framework: the first model resolves the discrete primitive category, and the second model concentrates on continuous attribute recovery. Such separation keeps each stage interpretable and makes it easier to diagnose whether an error originates from type prediction or from attribute reasoning.

Inference efficiency. Table 7 summarizes runtime and accuracy under different precision settings and inference frameworks. It compares HuggingFace inference with vLLM and reports whether lower-precision execution changes classification quality. These results are placed in the appendix because they mainly affect deployment choices rather than the core modeling contribution. The benchmark is included to clarify that memory reduction and runtime acceleration are not always coupled. Quantized execution depends on kernel support, batching behavior, and framework-level optimizations, so the same precision setting can lead to different runtime behavior across inference backends.

Table 7: Quantization and inference framework benchmark.

Precision	Runtime	ACC (Primitives)	Macro-F1	Weighted-F1
HuggingFace - Llama 3.1 8B				
FP16	9.86h	0.9361	0.8699	0.9344
INT8	33.52h	0.9357	0.8696	0.9343
INT4	10.49h	0.9349	0.8673	0.9331
vLLM - Llama 3.1 8B				
FP16	6.97h	0.9361	0.8698	0.9344
FP8	3.87h	0.9360	0.8695	0.9343
vLLM - Gemma 2-2B				
FP16	5.54h	0.9349	0.8673	0.9330
vLLM - Phi 3.5 mini - 3.8B				
FP16	6.38h	0.9229	0.8412	0.920

The quantization results show that lower precision does not automatically reduce runtime under every framework. In the HuggingFace setting, FP16 remains faster than INT8 and INT4, while vLLM with FP8 provides a clear runtime advantage with almost unchanged accuracy. This supports using vLLM FP8 when inference efficiency is the priority. The near-identical accuracy across FP16 and FP8 further suggests that the missing-primitive classification stage is robust to moderate numerical

approximation. For resource-constrained CAD assistance systems, this makes low-precision vLLM inference a practical option without changing the modeling pipeline.