International Journal Science and Technology
IJST Vol 5 No. 3 | November 2026 | ISSN: 2828-7223 (print), ISSN: 2828-7045 (online), Page 369-391

Comparative Evaluation of TOON and JSON Serialization Formats for Token Efficiency, Answer Accuracy, and Structural Reliability in DeepSeek

Syahrizal Andhika*, Muhammad Shafari Rahmat
Information System, Gunadarma University, Depok, Indonesia

Article History
Received : August 18 , 2026
Revised : September 17, 2026
Accepted : October 18, 2026
Published : October 25, 2026

Corresponding author*:
syahrizal@staff.gunadarma.ac.id

Cite This Article [*APA Style*]:
Andhika, S., & Rahmat, M. S. (2026). Comparative Evaluation of TOON and JSON Serialization Formats for Token Efficiency, Answer Accuracy, and Structural Reliability in DeepSeek. *International Journal Science and Technology*, *5*(3), 369–391.

DOI:
https://doi.org/10.56127/ijst.v5i3.2669

Abstract: The increasing adoption of Large Language Models (LLMs) requires efficient structured-data serialization to reduce token consumption and operational costs without compromising accuracy. Token-Oriented Object Notation (TOON) offers an alternative to JavaScript Object Notation (JSON), but its performance on DeepSeek remains insufficiently evaluated. **Objective**: This study evaluates token efficiency, answer accuracy, structural validation reliability, and estimated API processing costs across six serialization formats. **Methods**: A descriptive comparative experiment employed 11 official benchmark datasets, five synthetic datasets, and 209 evaluation questions. A total of 1,154 DeepSeek API requests were performed, while offline token consumption was measured separately using the o200k_base tokenizer. **Results**: TOON reduced token consumption by 58.8% for flat datasets and 21.9% for mixed-structure datasets compared with pretty-printed JSON. TOON achieved the highest observed answer accuracy (96.2%), compared with JSON (95.2%), although structural validation accuracy remained at 60.0%. **Implications**: The findings support workload-dependent serialization selection and emphasize the importance of deterministic schema validation and output-token monitoring for reliable, cost-efficient LLM applications. **Originality**: This study extends the official TOON benchmarking framework to DeepSeek and integrates token efficiency, accuracy, structural reliability, and cost evaluation, demonstrating that compression advantages do not necessarily correspond to improved accuracy or lower processing costs.

Keywords: Answer Accuracy; DeepSeek; JSON; Token Efficiency; Token-Oriented Object Notation

INTRODUCTION

The operational cost and performance of applications integrating Large Language Models (LLMs) through Application Programming Interfaces (APIs) are significantly influenced by the number of tokens processed during inference. Structured data serialization represents a critical technical factor because syntactic elements, repeated field names, and hierarchical representations contribute directly to input token consumption and computational expenditure. JavaScript Object Notation (JSON), standardized under RFC

© 2026 by the author. This is an open-access article under the Creative Commons Attribution ShareAlike license (CC BY SA)

8259, is widely adopted for structured data interchange because of its interoperability, readability, and machine-processable representation (Bray, 2017). However, its conventional syntax was primarily designed for data exchange rather than token-efficient communication with language models. Token-Oriented Object Notation (TOON), introduced in 2025, addresses this limitation by representing uniform arrays of objects through a compact tabular notation that minimizes repetitive structural characters (toon, 2025b). The technical significance of this approach is reflected in the experimental measurements obtained in this study, where TOON reduced token consumption by 58.8% compared with pretty-printed JSON for flat datasets and by 21.9% for mixed-structure datasets. Nevertheless, reductions in token consumption do not necessarily guarantee improvements in overall system performance because serialization formats may also affect answer accuracy, structural interpretation, output token generation, and API processing costs. Consequently, evaluating serialization efficiency requires an integrated assessment of measurable technical indicators rather than relying exclusively on input token reduction.

Existing research on structured representations for LLM applications can be classified into three principal technical approaches. The first approach focuses on experimental benchmarking of serialization formats and their computational efficiency. The official TOON benchmark reports substantial token savings for tabular datasets (toon, 2025a) while (Kutschka & Geiger, 2026) identified token reductions of up to 18% accompanied by accuracy degradation of approximately nine percentage points and limitations in multi-turn parsing and parallel tool-call scenarios. (Matveev, 2026) further demonstrated that TOON efficiency exhibits nonlinear behavior associated with dataset size and constrained decoding conditions. The second approach investigates the influence of representation constraints and structured information encoding on LLM reasoning performance. (Tam et al., 2024) demonstrated that output-format restrictions can influence model performance, whereas (Markowitz et al., 2025) showed that different textual representations of knowledge graphs affect reasoning accuracy. These findings suggest that data representation influences not only computational efficiency but also the model's ability to interpret and process structured information, although the experimental settings differ from input serialization benchmarks. The third approach emphasizes prompt compression and computational optimization. (Jiang et al., 2023) introduced LLMLingua as a prompt compression method intended to reduce inference costs, while (Li et al., 2025) examined compression strategies and their associated trade-offs between token reduction and model

performance. Despite these developments, the studies reviewed in the original manuscript have not established whether TOON's reported efficiency and accuracy characteristics are reproducible on DeepSeek. Furthermore, the relationships among structural uniformity, token savings, answer accuracy, structural validation, and cache-dependent processing costs remain insufficiently investigated for this model. This research gap highlights the need for a controlled experimental comparison of TOON and JSON using consistent benchmark datasets and quantitative performance indicators specifically evaluated on DeepSeek.

This study aims to experimentally investigate and quantitatively compare the performance of TOON and JSON serialization formats in DeepSeek-based structured data processing. Specifically, the research evaluates differences in token consumption across flat, nested, semi-uniform, and deeply structured datasets; measures answer accuracy using standardized benchmark questions; assesses the reliability of structural interpretation and incomplete-data detection; and examines the influence of dataset uniformity on the relative efficiency of each serialization format. The experimental framework employs eleven official TOON accuracy benchmark datasets, five additional synthetic datasets for offline token-efficiency measurement, and 209 benchmark questions evaluated across six serialization formats, including TOON, pretty-printed JSON, compact JSON, YAML, XML, and CSV where structurally applicable. Token efficiency is measured using the o200k_base tokenizer for offline serialization analysis, whereas actual DeepSeek API token-usage records are used to evaluate processing costs and input token consumption during accuracy testing. The study additionally compares DeepSeek-specific results with published benchmark outcomes from four other LLMs to examine cross-model consistency without conducting additional experiments on those models. Furthermore, API processing costs are estimated under cache-hit and cache-miss scenarios to determine whether reductions in input token consumption translate into lower total operational costs. Through these measurements, the research seeks to establish technically grounded criteria for selecting serialization formats according to dataset characteristics, accuracy requirements, structural reliability, and computational efficiency.

The principal technical contribution of this research is an independent replication and extension of the official TOON benchmarking framework through the inclusion of DeepSeek as an additional evaluation model and the integration of multiple engineering performance indicators. Unlike previous studies that primarily examined token

compression, constrained decoding, or representation effects on selected language models, this investigation combines token-efficiency measurement, answer-accuracy evaluation, structural validation, dataset-structure classification, and API cost estimation within a unified experimental framework. The underlying engineering hypothesis is that TOON can reduce input token consumption relative to conventional pretty-printed JSON while maintaining comparable or improved answer accuracy, although the magnitude of these advantages depends on structural uniformity, serialization complexity, and output token requirements. The experimental findings indicate that TOON achieves the highest observed answer accuracy on DeepSeek (96.2%) while reducing input token consumption, with particularly notable accuracy improvements in nested and semi-uniform datasets. However, the results also reveal limitations in detecting incomplete structured data and demonstrate that minimizing input tokens does not necessarily minimize total API costs under different caching conditions. These findings contribute empirical evidence regarding the relationship between structured data representation and LLM inference performance, while extending existing knowledge of the nonlinear efficiency characteristics of token-oriented serialization formats. From an engineering perspective, the study provides practical criteria for selecting serialization strategies in DeepSeek-based applications, particularly when balancing token efficiency, answer accuracy, structural reliability, and operational cost. Nevertheless, because each experimental configuration was evaluated through a single run, the reported differences should be interpreted as descriptive findings rather than statistically established performance superiority.

RESEARCH METHOD

The unit of analysis in this study was the structured data serialization and processing system used to transmit contextual information to the DeepSeek Large Language Model (LLM) through an Application Programming Interface (API). The engineering object under investigation consisted of six serialization formats, namely Token-Oriented Object Notation (TOON), pretty-printed JavaScript Object Notation (JSON), compact JSON, YAML, XML, and CSV, with CSV evaluated only for datasets supporting fully tabular representations. The system was examined in terms of input token consumption, answer accuracy, structural validation reliability, and estimated API processing costs. Dataset structural characteristics, including uniform tabular arrangements, nested objects, semi-uniform records, time-series data, and deeply hierarchical configurations, were considered

operational factors potentially influencing serialization efficiency and model interpretation. The experimental environment employed the official toon-format/toon benchmarking harness extended to support the DeepSeek API through the Vercel AI SDK, with deepseek-v4-flash serving as the primary evaluation model. The investigation focused on the relationship between serialization format, dataset structure, and measurable LLM processing performance rather than the development of a complete production application, such as a Retrieval-Augmented Generation (RAG) system or an autonomous agent. This unit of analysis enabled a direct technical comparison of serialization alternatives under a common benchmarking framework.

The research employed a quantitative experimental-comparative design using a replication-with-extension approach based on the official TOON benchmarking methodology (toon-format, 2025b). The experimental extension involved integrating DeepSeek as an additional model provider while maintaining the benchmark's established datasets, serialization procedures, and automated answer-validation framework. The independent experimental factor was the serialization format, whereas the principal response variables were token consumption, answer accuracy, structural validation performance, and estimated API processing cost. Dataset structure was examined as an additional classification factor to evaluate variations in serialization performance across different data characteristics. The experimental procedure consisted of two separate measurement pathways. The first pathway evaluated serialization efficiency using five larger synthetic datasets without invoking the DeepSeek API, whereas the second pathway evaluated answer accuracy and actual token usage using eleven official benchmark datasets processed through the API. Each eligible combination of dataset, serialization format, and benchmark question was executed once, producing 1,154 API requests for the accuracy and cost evaluations. This single-run configuration was suitable for descriptive benchmarking and identifying observable performance differences, but it did not support statistical inference regarding the significance or repeatability of the measured differences. Accordingly, the study employed descriptive comparative analysis rather than inferential statistical testing.

The research data were obtained from the publicly available official TOON benchmark repository and experimental records generated during the DeepSeek evaluation. Two dataset groups were used for different measurement purposes. The accuracy evaluation group comprised eleven benchmark datasets, including 100 uniform employee records, 50

nested e-commerce orders, 60 days of time-series analytics data, 100 GitHub repository records, 75 semi-uniform event logs, one deeply nested configuration dataset, and five additional structural validation datasets. These datasets were selected to represent different structural characteristics while maintaining manageable API processing costs. The token-efficiency evaluation group comprised five larger synthetic datasets containing 2,000 employee records, 500 e-commerce orders, 365 days of analytics data, 100 GitHub repository records, and 2,000 event logs. The accuracy benchmark included 209 predefined questions per applicable serialization format, with CSV restricted to questions involving structurally compatible datasets. The sampling strategy followed total sampling of the selected official benchmark datasets rather than random sampling from a broader population of structured data. Primary experimental records consisted of serialized payloads, model-generated answers, correctness indicators, and API-reported input and output token usage. Secondary technical data included published benchmark results for Claude Haiku, Gemini Flash, GPT-5 Nano, and Grok Fast, obtained from the official TOON benchmark repository (toon-format, 2025b). These secondary results were used exclusively for cross-model comparison and were not generated through additional experiments conducted by the authors. All benchmark datasets were publicly available and did not contain personal information, while DeepSeek API usage was conducted in accordance with the provider's applicable terms of service.

Data collection was conducted using an automated TypeScript/Node.js benchmarking harness incorporating the official TOON serialization procedures, the Vercel AI SDK, and the DeepSeek API client. The experimental workflow began with dataset preparation and classification according to structural characteristics, followed by automatic encoding into the six serialization formats wherever structurally applicable. For the offline token-efficiency evaluation, each serialized payload was processed using the o200k_base tokenizer to calculate token consumption without generating API requests. This tokenizer was used as a consistent cross-format measurement proxy and was not assumed to represent the native tokenization behavior of DeepSeek. For the accuracy evaluation, the serialized datasets were incorporated into benchmark prompts containing predefined questions and submitted to the deepseek-v4-flash model through the API. Each request was executed once, and the automated logging system recorded the model-generated answer, its correctness status against the benchmark ground truth, and the actual usage.inputTokens and usage.outputTokens values reported by the API. Answer correctness was evaluated

using automated type-aware matching against predefined reference answers without employing an additional LLM-based evaluator. Structural reliability was examined through benchmark questions designed to assess the model's ability to identify valid records, truncated arrays, additional rows, inconsistent column counts, and missing required fields. The same benchmarking framework was used across applicable formats to support procedural consistency, while the offline tokenizer measurements and actual API usage measurements were maintained as separate datasets to prevent methodological conflation. No physical test rig or sensor calibration was required because the experiment investigated software-based serialization and inference behavior. The supplied experimental records did not establish repeated-run calibration, deterministic model settings, or controlled API cache states; therefore, these conditions were not assumed to have been controlled.

Data analysis was performed using descriptive statistical measures and comparative performance indicators calculated for each serialization format and dataset category. Token efficiency was quantified by comparing the number of tokens required to represent equivalent structured datasets, using pretty-printed JSON as the primary reference format. The relative token reduction was calculated using Equation (1), where T_{JSON} represents the token count for pretty-printed JSON and T_f represents the token count for the evaluated serialization format.

$$R_T = \frac{T_{\mathrm{JSON}} - T_f}{T_{\mathrm{JSON}}} \times 100\%$$

Answer accuracy was determined from the proportion of benchmark questions answered correctly relative to the total number of applicable questions for each format, as expressed in Equation (2), where N_c denotes the number of correct answers and N_q denotes the number of evaluated questions.

$$A = \frac{N_c}{N_q} \times 100\%$$

Structural validation performance was evaluated using the same correctness-based principle, restricted to the structural validation question category. For comparisons of answer accuracy between TOON and JSON, the absolute difference was expressed in percentage points to distinguish accuracy changes from relative percentage changes. The combined relationship between accuracy and serialized payload length was additionally

evaluated using an accuracy-per-thousand-token indicator, as defined in Equation (3), where $\overline{T}_f$ denotes the average serialized payload token count for format f, measured using the offline tokenizer.

$$E_f = \frac{A_f}{\overline{T}_f/1000}$$

API processing costs were estimated using the actual input and output token usage reported by DeepSeek, multiplied by the corresponding published token pricing rates. The estimated processing cost for 1,000 queries was calculated using Equation (4), where $\overline{T}_{in}$ and $\overline{T}_{out}$ represent average input and output tokens per query, respectively, andP_{in} and P_{out}represent the corresponding prices per one million tokens.

$$C_{1000} = 1000\left(\frac{\overline{T}_{in}P_{in} + \overline{T}_{out}P_{out}}{10^6}\right)$$

Cost calculations were conducted under two bounding scenarios, cache hit and cache miss, using the published DeepSeek pricing information because individual request-level cache status was not available in the experimental logs. The resulting token-efficiency, accuracy, structural validation, and cost indicators were compared across serialization formats and dataset categories to identify observable performance patterns associated with structural uniformity. DeepSeek accuracy rankings were also compared descriptively with the published results of four other LLMs from the official benchmark repository, without treating those secondary results as independently replicated experiments. Python was used for additional data aggregation and cost calculations, while the TypeScript/Node.js harness supported experimental data acquisition. No paired t-test, Wilcoxon signed-rank test, or other inferential significance test was conducted because each experimental configuration was evaluated only once. Furthermore, some structural validation cases contained only one question per format, limiting the reliability of conclusions concerning structural error detection. The results were therefore interpreted as descriptive and exploratory rather than statistically conclusive. Generalization was additionally constrained by the use of official TOON benchmark datasets, the absence of Indonesian-language or domain-specific datasets, differences in size between the accuracy and offline efficiency datasets, and the use of o200k_base as a tokenizer proxy rather than a DeepSeek-native tokenizer.

RESULT

The experimental results were obtained from two complementary evaluation pathways: offline token-efficiency measurement using five synthetic datasets and answer-accuracy evaluation using eleven official TOON benchmark datasets processed through the DeepSeek API. The evaluation covered six serialization formats, namely TOON, pretty-printed JSON, compact JSON, YAML, XML, and CSV, with CSV restricted to structurally compatible datasets. The results are presented in accordance with the research objectives, beginning with token consumption, followed by answer accuracy, structural validation reliability, dataset-dependent performance, and estimated API processing costs. Offline token measurements and actual API token usage are reported separately because they were obtained using different measurement procedures.

Token Consumption and Serialization Efficiency

The first evaluation examined differences in token consumption across serialization formats using five synthetic datasets containing 2,000 employee records, 500 e-commerce orders, 365 days of analytics data, 100 GitHub repository records, and 2,000 event logs. Token counts were obtained using the o200k_base tokenizer applied directly to the serialized payloads. The datasets were grouped into two structural categories: flat-only datasets, which support uniform tabular representations, and mixed-structure datasets, which contain nested or semi-uniform data. Table 1 presents the aggregated token consumption for each serialization format, using pretty-printed JSON as the reference.

Table 1. Comparison of Token Consumption Across Serialization Formats and Dataset Structures

Serialization Format	Flat-Only (Tokens)	Difference vs. JSON	Mixed-Structure (Tokens)	Difference vs. JSON
CSV	64,041	−61.1%	N/A	N/A
TOON	67,834	−58.8%	227,896	−21.9%
JSON compact	104,720	−36.3%	198,560	−31.9%
YAML	131,028	−20.3%	241,459	−17.2%
JSON (pretty)	164,448	Reference	291,620	Reference
XML	190,316	+15.7%	330,065	+13.1%

Source: Experimental benchmark harness data (2026), adapted from the official toon-format/toon framework. Token counts were obtained using o200k_base rather than a DeepSeek-native tokenizer.

The results reveal three dominant patterns. First, TOON achieved substantial token reduction in flat-only datasets, requiring 67,834 tokens compared with 164,448 tokens for pretty-printed JSON, corresponding to a reduction of 58.8%. CSV achieved the lowest token consumption in this category at 64,041 tokens, approximately 5.9% fewer than TOON. Second, the relative efficiency of TOON decreased in mixed-structure datasets, where its token consumption reached 227,896 tokens, representing a 21.9% reduction relative to pretty-printed JSON. Third, compact JSON outperformed TOON in mixed-structure datasets, requiring 198,560 tokens, or approximately 12.9% fewer tokens than TOON. XML consistently exhibited the highest token consumption in both structural categories.

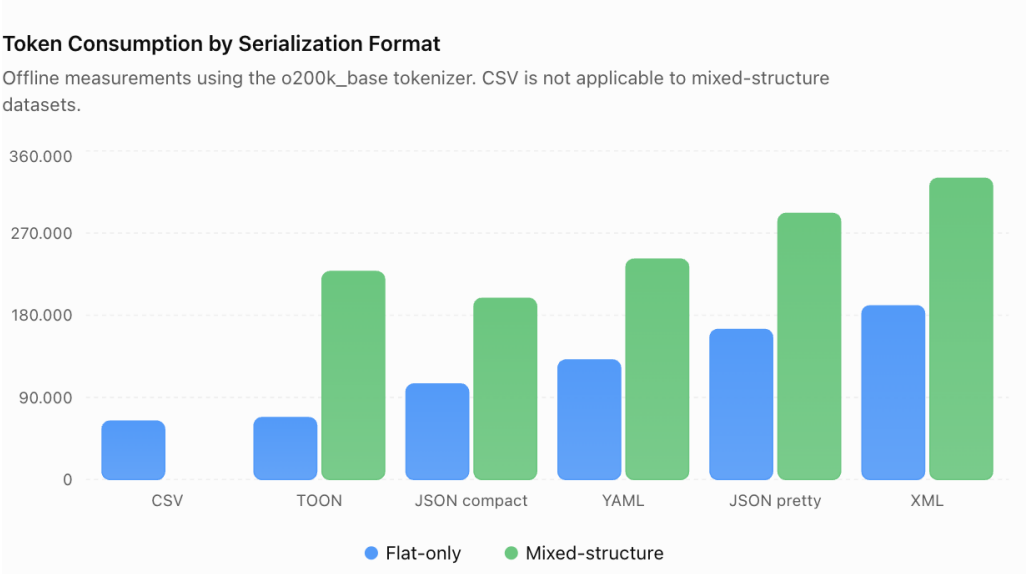

Figure 1. Token Consumption

These findings indicate that serialization efficiency depends strongly on dataset structure. TOON benefits from uniform tabular arrangements because repeated field names and structural delimiters can be represented more compactly. However, nested and semi-uniform datasets reduce this advantage because additional structural representations are required. The results therefore establish that TOON is more token-efficient than pretty-printed JSON across the evaluated dataset categories, but it is not consistently more efficient than compact JSON or CSV.

Answer Accuracy and Token Efficiency on DeepSeek

The second evaluation investigated answer accuracy using eleven official benchmark datasets and 209 predefined questions per applicable serialization format. Model responses were evaluated through automated type-aware matching against benchmark ground-truth answers. Table 2 presents the observed answer accuracy, average serialized payload token count, and accuracy-per-thousand-token efficiency for each format.

Table 2. Answer Accuracy and Token Efficiency of Serialization Formats on DeepSeek

Format	Answer Accuracy	Average Payload Tokens	Accuracy per 1,000 Tokens
TOON	96.2% (201/209)	2,734	35.2
JSON (pretty)	95.2% (199/209)	4,538	21.0
JSON compact	94.7% (198/209)	3,072	30.8
XML	94.3% (197/209)	5,162	18.3
CSV	93.6% (102/109)	N/A	N/A
YAML	92.3% (193/209)	3,716	24.9

Source: Experimental benchmark harness data (2026). CSV was applicable to only 109 of the 209 questions. Average payload token counts were calculated using o200k_base and exclude additional API prompt components.

TOON achieved the highest observed answer accuracy at 96.2%, corresponding to 201 correct answers from 209 benchmark questions. This result exceeded pretty-printed JSON by 1.0 percentage point, compact JSON by 1.5 percentage points, XML by 1.9 percentage points, and YAML by 3.9 percentage points. CSV achieved 93.6% accuracy; however, its evaluation involved only 109 structurally compatible questions, making direct comparison with the full evaluation set inappropriate.

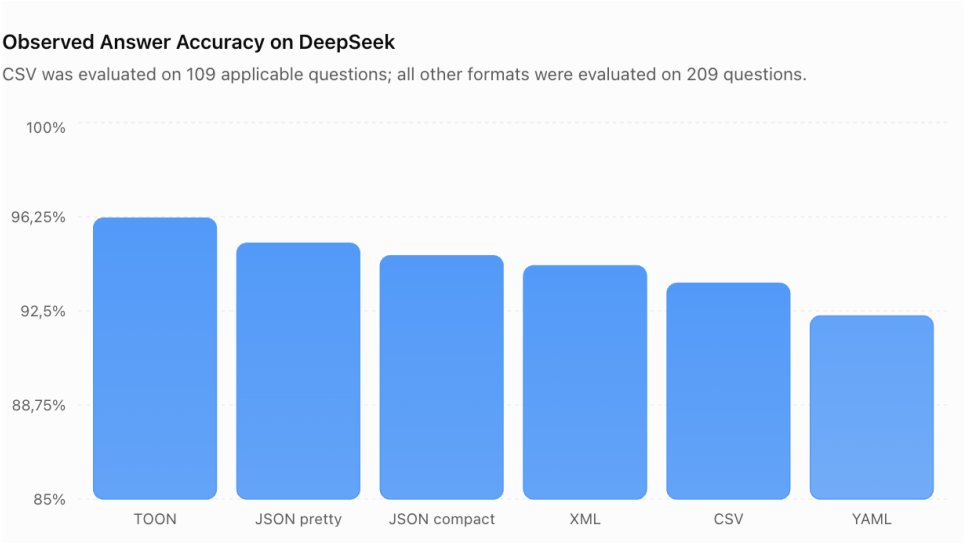

Figure 2. Accuracy on DeepSeek

TOON also exhibited the highest accuracy-per-thousand-token efficiency score at 35.2, compared with 21.0 for pretty-printed JSON, corresponding to a relative improvement of approximately 67.6%. Compact JSON ranked second on this metric with a score of 30.8. These results indicate that, within the evaluated benchmark, TOON combined relatively low serialized payload token consumption with high observed answer accuracy. Nevertheless, the accuracy differences were descriptive because each experimental configuration was evaluated only once.

To examine whether the observed ranking was consistent across language models, DeepSeek results were compared with published benchmark outcomes from four additional LLMs. These secondary results were obtained from the official TOON benchmark repository and were not independently reproduced in the present study.

Table 3. Highest-Accuracy Serialization Formats Across Five Large Language Models

Model	Best Format	Highest Accuracy	TOON Accuracy	Difference from Best
deepseek-v4-flash	TOON	96.2%	96.2%	0.0 pp
gemini-3-flash-preview	XML	98.1%	96.7%	−1.4 pp
gpt-5-nano	TOON / JSON compact	90.9%	90.9%	0.0 pp
claude-haiku-4-5	TOON	59.8%	59.8%	0.0 pp
grok-4-1-fast-non-reasoning	TOON	58.4%	58.4%	0.0 pp

Source: DeepSeek results from the authors' experiment (2026); other model results from the official toon-format benchmark (2025b). pp denotes percentage points.

TOON achieved the highest accuracy or shared the highest rank in four of the five evaluated models. The exception was Gemini Flash, where XML achieved 98.1% accuracy compared with 96.7% for TOON. The results also demonstrate substantial differences in absolute benchmark accuracy among models, ranging from 58.4% for Grok to 98.1% for Gemini. Because the benchmark used strict automated answer matching, these absolute differences should not be interpreted as general measures of model capability. The technically relevant comparison is the relative performance of serialization formats within each model.

Structural Validation Reliability

Structural validation reliability was evaluated using five benchmark cases designed to assess whether the model could identify valid and invalid structured datasets. The test cases included complete valid data, truncated arrays, additional rows, inconsistent column counts, and missing required fields. Each case contained one question per serialization format, resulting in a limited evaluation sample. Table 4 summarizes the correctness of the model's structural validation responses.

Table 4. Structural Validation Results Across Serialization Formats on DeepSeek

Validation Case	TOON	JSON Pretty	JSON Compact	YAML	XML	CSV
Complete valid data	Correct	Correct	Correct	Correct	Incorrect	Incorrect
Truncated array	Incorrect	Incorrect	Incorrect	Incorrect	Correct	Correct
Additional rows	Correct	Correct	Correct	Incorrect	Correct	Correct
Inconsistent column count	Correct	Correct	Correct	Correct	Correct	Correct
Missing required field	Incorrect	Incorrect	Incorrect	Incorrect	Correct	Correct

Source: Experimental benchmark harness data (2026). Each validation case contained one question per format (N = 1).

The structural validation results demonstrate an asymmetric error pattern among serialization formats. TOON, pretty-printed JSON, compact JSON, and YAML correctly identified complete valid datasets but failed to identify truncated arrays and missing required fields. Conversely, XML and CSV correctly identified both incomplete-data cases but incorrectly classified the complete valid dataset. All six formats correctly identified inconsistent column counts, while YAML was the only format that incorrectly evaluated the additional-row case.

To provide a broader assessment of model behavior, accuracy was also examined across five benchmark question categories: field retrieval, aggregation, filtering, structural awareness, and structural validation. The results are presented in Table 5.

Table 5. Answer Accuracy by Benchmark Question Category on DeepSeek

Question Category	TOON	JSON Pretty	JSON Compact	XML	CSV	YAML
Field retrieval	97.1%	95.6%	92.6%	94.1%	100.0%	92.6%
Aggregation	98.4%	96.8%	98.4%	95.2%	89.7%	93.7%
Filtering	97.9%	100.0%	97.9%	97.9%	100.0%	100.0%
Structural awareness	92.0%	88.0%	92.0%	88.0%	81.3%	84.0%
Structural validation	60.0%	60.0%	60.0%	80.0%	80.0%	40.0%

Source: Experimental benchmark harness data (2026). Question-category sample sizes differ according to format applicability.

TOON achieved high accuracy in field retrieval (97.1%), aggregation (98.4%), filtering (97.9%), and structural awareness (92.0%). However, its structural validation accuracy was limited to 60.0%, equivalent to pretty-printed and compact JSON. XML and CSV achieved higher structural validation accuracy at 80.0%, while YAML recorded the lowest value at 40.0%. These results suggest that strong performance in information retrieval and aggregation does not necessarily correspond to reliable identification of structural inconsistencies. Because the structural validation category contained only five questions, the observed differences should be interpreted as preliminary indicators rather than conclusive evidence of format-specific reliability.

Influence of Dataset Structure on Serialization Performance

The relationship between dataset structure and serialization performance was examined by classifying benchmark datasets according to tabular eligibility, defined as the proportion of the dataset structure that could be represented in a uniform tabular arrangement. The analysis compared TOON and pretty-printed JSON in terms of answer accuracy and relative token consumption across six dataset categories. Table 6 presents the corresponding results.

Table 6. Comparison of TOON and JSON Performance Across Dataset Structures on DeepSeek

Dataset	Tabular Eligibility	TOON Accuracy	JSON Accuracy	Accuracy Difference	TOON Token Reduction
Employee records (uniform)	100%	97.6%	100.0%	−2.4 pp	60.6%
Analytics time-series (uniform)	100%	100.0%	100.0%	0.0 pp	57.9%
GitHub repositories (uniform, long text)	100%	90.9%	90.9%	0.0 pp	42.1%
E-commerce orders (nested)	33%	100.0%	95.1%	+4.9 pp	32.7%
Event logs (semi-uniform)	50%	96.7%	90.0%	+6.7 pp	14.6%
Nested configuration (DeepSeek)	0%	96.6%	100.0%	−3.4 pp	28.9%

Source: Experimental benchmark harness data (2026). Accuracy differences are expressed in percentage points relative to pretty-printed JSON.

The results indicate that the highest token savings were obtained from datasets with fully uniform tabular structures. Employee records and analytics time-series datasets achieved token reductions of 60.6% and 57.9%, respectively, whereas semi-uniform event logs achieved a smaller reduction of 14.6%. However, the relationship between tabular eligibility and answer accuracy followed a different pattern. TOON did not improve accuracy relative to JSON for the three fully tabular datasets. Instead, the largest positive accuracy differences were observed in nested e-commerce orders (+4.9 percentage points) and semi-uniform event logs (+6.7 percentage points).

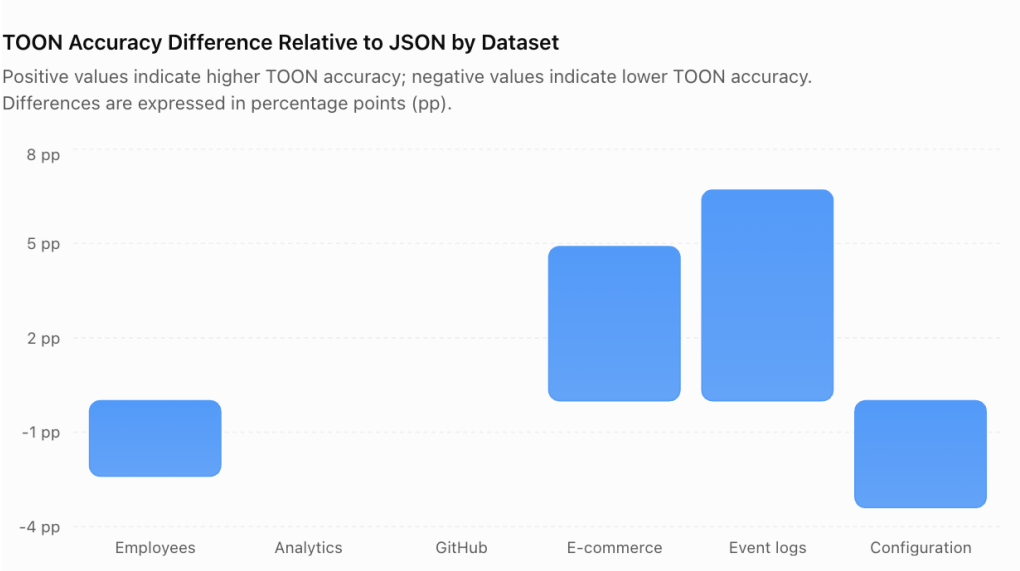

Figure 3. TOON Answer Accuracy Difference Relative to Pretty-Printed JSON Across Dataset Types on DeepSeek.

An additional observation concerns the GitHub repository dataset, which contained long textual fields despite having 100% tabular eligibility. TOON required 8,779 tokens compared with 8,527 tokens for CSV, indicating that quotation and character-escaping requirements can reduce its efficiency for text-intensive datasets. Overall, these findings demonstrate that structural uniformity strongly influences token savings but does not produce a corresponding monotonic improvement in answer accuracy. The observed accuracy advantage of TOON was concentrated in selected nested and semi-uniform datasets rather than uniformly structured datasets.

Estimated API Processing Costs

The final evaluation estimated DeepSeek API processing costs using actual input and output token usage recorded during the accuracy benchmark. Cost calculations employed the pricing values reported in the original experimental draft for July 15, 2026: USD 0.14 per million input tokens for cache misses, USD 0.0028 per million input tokens for cache hits, and USD 0.28 per million output tokens. Two bounding scenarios were considered because request-level cache status was not recorded. The average input and output token counts per query were 4,655 and 1,235 for TOON; 5,316 and 898 for compact JSON; 7,651 and 1,352 for pretty-printed JSON; 6,315 and 1,107 for YAML; 8,752 and 800 for XML; and 4,040 and 919 for CSV. These API-reported input token counts include the benchmark instructions and question text and therefore differ from the serialized payload measurements presented in Table 2.

Table 7. Estimated DeepSeek API Processing Costs per 1,000 Queries

Format	Cache Miss (USD)	Cache Miss (IDR)	Cache Hit (USD)	Cache Hit (IDR)
CSV	0.8230	14,866	0.2687	4,854
JSON compact	0.9957	17,985	0.2663	4,810
TOON	0.9974	18,016	0.3588	6,481
YAML	1.1940	21,567	0.3276	5,917
XML	1.4493	26,179	0.2484	4,487
JSON (pretty)	1.4497	26,186	0.4000	7,225

Source: Experimental benchmark harness data (2026) and DeepSeek pricing information reported for July 15, 2026. Currency conversion uses USD 1 = IDR 18,063, as stated in the original manuscript. Values are estimates, not observed billing charges.

Under the cache-miss scenario, TOON exhibited an estimated cost of USD 0.9974 per 1,000 queries, approximately 31.2% lower than pretty-printed JSON at USD 1.4497. CSV and compact JSON achieved marginally lower estimated costs than TOON, at USD 0.8230 and USD 0.9957, respectively. However, the relative cost ranking changed under the cache-hit scenario. XML achieved the lowest estimated cost at USD 0.2484 per 1,000 queries, followed by compact JSON at USD 0.2663 and CSV at USD 0.2687. TOON increased to USD 0.3588, making it the second most expensive format after pretty-printed JSON.

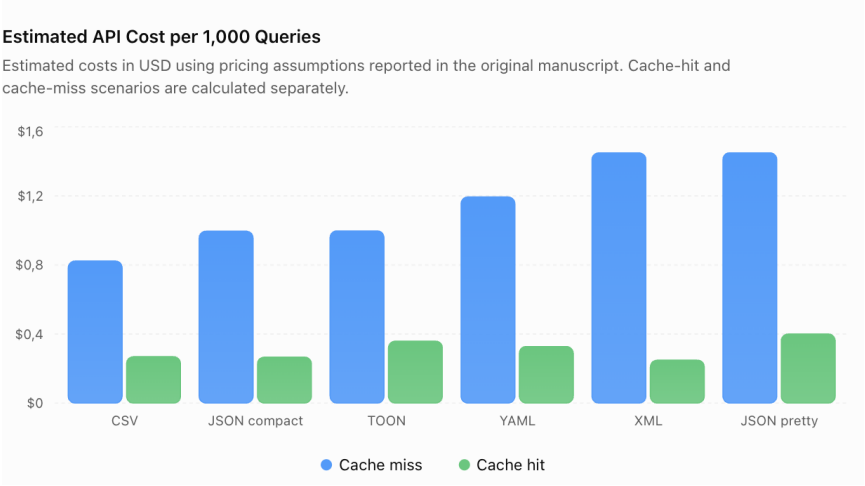

Figure 4. Estimated API Cost

The cost results reveal that lower input token consumption does not necessarily correspond to lower total API processing costs. Although TOON substantially reduced input token usage relative to pretty-printed JSON, its average output token consumption remained comparatively high at 1,235 tokens per query. Conversely, XML generated the lowest average output token count at 800 tokens per query despite having the highest input token consumption. Consequently, XML became the least expensive format in the cache-hit scenario, where input token prices were substantially reduced. These findings establish that total API cost is jointly influenced by input token consumption, output token generation, and caching assumptions rather than serialization efficiency alone.

DISCUSSION

The experimental findings demonstrate that serialization format selection influences multiple dimensions of Large Language Model (LLM) processing performance, including token efficiency, answer accuracy, structural validation reliability, and API processing costs. TOON generally provided a more compact representation than pretty-printed JSON while maintaining competitive answer accuracy on DeepSeek. However, its performance advantages were not uniform across dataset structures or evaluation metrics. Uniform tabular datasets produced the greatest token savings, whereas the most favorable accuracy differences emerged from selected nested and semi-uniform datasets. Furthermore, improvements in token efficiency did not consistently translate into better structural error detection or lower total processing costs. These findings indicate that serialization

performance should be evaluated as a multidimensional engineering optimization problem rather than a single-objective token minimization task.

The observed token-efficiency differences can be explained by the structural encoding mechanisms employed by each serialization format. Pretty-printed JSON repeatedly represents field names, quotation marks, delimiters, and hierarchical structures, increasing the number of tokens required to encode large collections of similar objects. TOON reduces this redundancy by representing uniform records through shared field definitions and compact tabular arrangements. Consequently, its compression advantage becomes more pronounced when datasets contain repeated attributes and consistent record structures. Conversely, nested objects, irregular fields, and semi-uniform records require additional structural information, reducing the efficiency of tabular encoding. This explains why compact JSON can outperform TOON in certain mixed-structure datasets despite TOON's substantial advantage over pretty-printed JSON. The reduced token savings observed in text-intensive GitHub repository data also suggest that long string values and character-escaping requirements can limit the benefits of structural compression. These results support the interpretation that serialization efficiency depends on the interaction between structural redundancy, data hierarchy, and tokenizer behavior rather than dataset size alone.

The relationship between serialization format and answer accuracy requires a different technical interpretation. Although TOON achieved the highest overall accuracy on DeepSeek, its largest advantages were observed in nested e-commerce and semi-uniform event-log datasets rather than fully tabular datasets. One possible explanation is that compact representation reduces repetitive syntactic information and improves the relative prominence of task-relevant values within the model's input context. However, this mechanism remains a hypothesis because the experiment did not directly measure attention allocation, internal model representations, or reasoning processes. The absence of a consistent accuracy advantage across all dataset categories indicates that token compression and semantic interpretation are related but distinct performance dimensions. Structural validation results reinforce this distinction. TOON performed well in information retrieval, aggregation, and structural awareness tasks but exhibited limitations in identifying truncated arrays and missing required fields. Because TOON represents certain structural relationships implicitly, incomplete records may not always produce sufficiently distinctive textual patterns for reliable detection by the model. Nevertheless, similar failures were observed in JSON and YAML, indicating that the limitation cannot

be attributed exclusively to TOON. Given the small number of structural validation questions, these findings should be interpreted as preliminary evidence requiring further experimental verification.

The findings provide a useful comparison with previous studies examining serialization efficiency and representation-dependent LLM performance. (Matveev, 2026)reported nonlinear efficiency characteristics of TOON associated with dataset size and constrained decoding conditions. The present study extends this perspective by demonstrating that structural uniformity and hierarchical complexity also influence token savings, suggesting that serialization efficiency cannot be predicted solely from the number of records. (Kutschka & Geiger, 2026) identified token reductions accompanied by accuracy degradation and limitations in more complex agentic processing scenarios. In contrast, the present DeepSeek evaluation showed that TOON achieved the highest observed answer accuracy among the evaluated formats. This difference does not invalidate the earlier findings because the studies employed different models, benchmark conditions, and evaluation tasks. Rather, it suggests that the relationship between token compression and model performance is dependent on the experimental configuration. Similarly, (Tam et al., 2024)demonstrated that formatting constraints can affect LLM performance, while (Markowitz et al., 2025) showed that textual representations of structured knowledge influence reasoning accuracy. The present results are consistent with the broader conclusion that representation design affects model behavior, although the current experiment does not establish the internal mechanisms responsible for these effects. Research on prompt compression by (Jiang et al., 2023; Li et al., 2025) further emphasizes the importance of balancing computational efficiency with model performance. Unlike general prompt compression techniques that modify or remove prompt content, TOON primarily reduces representational redundancy while retaining structured information. This distinction is relevant because serialization optimization can potentially improve efficiency without intentionally discarding task-relevant data, although semantic equivalence and output reliability must still be validated.

From an engineering perspective, the results suggest that serialization format selection should be treated as a workload-dependent design decision. For applications processing large collections of uniform records, TOON offers a practical mechanism for reducing input token consumption while maintaining competitive answer accuracy. However, compact JSON may be more suitable for irregular or deeply nested structures when

minimum token consumption and compatibility with existing software interfaces are prioritized. The structural validation findings also indicate that LLM-generated assessments should not replace deterministic schema validation. Applications requiring reliable detection of missing fields, inconsistent records, or incomplete payloads should implement validation procedures before data are submitted to the model. Such procedures may include JSON Schema validation, explicit field-count verification, required-attribute checks, and serialization round-trip testing. These safeguards are particularly relevant in engineering applications where incorrect interpretation of structured information could compromise downstream calculations, automated decisions, or system reliability. The proposed validation measures represent engineering recommendations derived from the observed limitations rather than additional procedures experimentally evaluated in this study.

The API cost analysis introduces another important consideration for system optimization. Although TOON reduced input token consumption relative to pretty-printed JSON, it did not consistently achieve the lowest estimated processing cost. This difference resulted from the combined influence of input tokens, output tokens, and the assumed cache-pricing scenario. Under cache-miss conditions, reducing input token consumption contributed substantially to lower estimated costs. Under cache-hit conditions, however, discounted input token pricing increased the relative importance of output token generation, allowing formats with larger input payloads but shorter model responses to become more economical. Therefore, serialization optimization should consider total token usage rather than input token reduction alone. For production systems, developers should evaluate representative workloads using actual API usage records, monitor input and output token distributions, and compare processing costs under realistic caching conditions. A technically appropriate selection procedure would prioritize structural compatibility and answer reliability before optimizing token consumption and cost. Furthermore, the selected serialization format should be evaluated periodically because changes in model behavior, tokenizer implementation, or API pricing may alter its relative advantages.

The principal novelty of this study lies in extending the official TOON benchmarking framework to DeepSeek and integrating token consumption, answer accuracy, structural validation, dataset characteristics, and estimated processing costs within a unified comparative evaluation. The findings contribute empirical evidence that the benefits of token-oriented serialization are conditional rather than universal. In particular, the

contrasting relationships between dataset uniformity and answer accuracy indicate that the structural characteristics producing the greatest token savings do not necessarily produce the greatest improvements in model responses. This distinction is relevant to the development of engineering criteria for serialization selection in LLM-based software architectures. Nevertheless, the single-run experimental design, limited structural validation samples, proxy tokenizer measurements, and absence of request-level cache information restrict the strength of causal and statistical conclusions. Future investigations should employ repeated trials, controlled model configurations, native tokenizer measurements, and expanded structural validation benchmarks to determine whether the observed differences remain consistent under broader operating conditions. Additional evaluations involving Indonesian-language datasets, domain-specific engineering information, Retrieval-Augmented Generation systems, and agentic workflows would further establish the practical applicability and generalizability of the proposed serialization selection approach.

CONCLUSION

This study evaluated the effects of structured data serialization formats on token efficiency, answer accuracy, structural validation reliability, and API processing costs using the DeepSeek large language model. The results demonstrated that TOON reduced token consumption by 58.8% for flat-only datasets and 21.9% for mixed-structure datasets compared with pretty-printed JSON. TOON achieved the highest observed answer accuracy of 96.2%, exceeding pretty-printed JSON by 1.0 percentage point. The greatest accuracy improvements were observed in event logs (+6.7 percentage points) and e-commerce datasets (+4.9 percentage points). However, structural validation accuracy remained at 60.0%, while processing costs varied according to input and output token consumption and caching conditions.

The primary scientific contribution of this research is the extension of the official TOON benchmarking framework to DeepSeek through an integrated evaluation of serialization efficiency, accuracy, structural reliability, and computational cost. The findings demonstrate that dataset structure is a critical factor influencing token efficiency, while improvements in token reduction do not necessarily correspond to higher accuracy or lower processing costs. These results provide empirical evidence and practical

considerations for selecting appropriate serialization formats in LLM-based applications, particularly when balancing computational efficiency, data interpretation accuracy, and operational requirements.

This study has several methodological limitations, including its single-run experimental design, limited structural validation samples, use of the o200k_base tokenizer as a proxy for DeepSeek tokenization, and the absence of request-level caching information. Consequently, the observed performance differences remain descriptive and cannot establish statistical significance or universal superiority across language models. Future research should incorporate repeated experimental trials, statistical significance testing, native tokenizer measurements, controlled caching conditions, and larger domain-specific datasets, including Indonesian-language data, to validate the findings and improve the generalizability of token-efficient serialization strategies.

Artificial Intelligence Use Statement

Artificial intelligence tools were used to assist with language refinement, grammatical correction, and stylistic editing during manuscript preparation. The DeepSeek large language model was employed as the experimental object and evaluated through a controlled benchmarking framework. The use of DeepSeek for experimental evaluation is distinct from the use of AI-assisted writing tools. The authors retain full responsibility for the accuracy, integrity, interpretation, and final content of the manuscript.

Author Contribution Statement

Syahrizal Andhika: Conceptualization, Methodology, Software, Investigation, Data curation, Formal analysis, Visualization, Writing – original draft. Muhammad Shafari Rahmat: Conceptualization, Methodology, Validation, Formal analysis, Writing – review & editing. All authors reviewed and approved the final version of the manuscript and agreed to be accountable for its accuracy, integrity, and scientific content.

Funding Statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflict of Interest Statement

The authors declare that they have no financial, professional, institutional, or personal conflicts of interest that could have influenced the design, conduct, analysis, interpretation, or reporting of this study.

REFERENCES

Bray, T. (2017). *The JavaScript Object Notation (JSON) Data Interchange Format* (RFC 8259, Issue. https://www.rfc-editor.org/rfc/rfc8259

DeepSeek. (2026). Models & Pricing. https://api-docs.deepseek.com/quick_start/pricing/

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., & Qiu, L. (2023). LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing,

Kutschka, L., & Geiger, B. (2026). *Notation Matters: A Benchmark Study of Token-Optimized Formats in Agentic AI Systems*. https://arxiv.org/abs/2605.29676

Li, Z., Liu, Y., Su, Y., & Collier, N. (2025). Prompt Compression for Large Language Models: A Survey. Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers),

Markowitz, E., Galiya, K., Ver Steeg, G., & Galstyan, A. (2025). *KG-LLM-Bench: A Scalable Benchmark for Evaluating LLM Reasoning on Textualized Knowledge Graphs*. https://arxiv.org/abs/2504.07087

Matveev, I. (2026). *Token-Oriented Object Notation vs JSON: A Benchmark of Plain and Constrained Decoding Generation*. https://arxiv.org/abs/2603.03306

Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H.-y., & Chen, Y.-N. (2024). Let Me Speak Freely? A Study On The Impact Of Format Restrictions On Large Language Model Performance. Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track,

toon, f. (2025a). *Benchmarks* (GitHub. https://github.com/toon-format/toon/tree/main/benchmarks

toon, f. (2025b). *TOON: Token-Oriented Object Notation*. In GitHub. https://github.com/toon-format/toon