Journal of Systems Engineering and Information Technology (JOSEIT)
Volume 4, No. 2, September (2025), Pages 51-56
e-ISSN: 2829-310X | 10.29207/joseit.

An Empirical Evaluation of ChatGPT as an Automated Machine Learning Code Generator for Image Classification

Dedi Mardianto[1*], Milla Apriliana[2], Eva Oktavia[3], Ideva Gaputra[4], Widya Wahyuni[5]
[12]Department of Information Technology, Politeknik Negeri Padang, Padang, Indonesia

[*]*dedimardianto@pnp.ac.id*

Abstract. The emergence of large language models such as ChatGPT has created unprecedented opportunities for automating software development processes, particularly within the machine learning domain. This study aims to empirically evaluate the effectiveness of ChatGPT in generating machine learning code for image classification tasks using the Keras framework. The research employs an experimental methodology utilizing the MNIST dataset, comprising 70,000 handwritten digit images. A systematic series of experiments was conducted through progressive prompting strategies, ranging from basic model construction to comprehensive evaluation protocols. The findings demonstrate that ChatGPT successfully generated 100% executable code without errors, with the resulting models achieving 99% accuracy on the test dataset. A notable discovery emerged in the form of "intelligent deviation" phenomena, wherein ChatGPT autonomously provided Convolutional Neural Network (CNN) architectures despite explicit requests for fully connected layers, demonstrating sophisticated contextual understanding. The generated code quality met professional standards with robust multi-library integration capabilities. This research provides the first systematic empirical contribution regarding large language model capabilities in machine learning code generation, offering significant implications for democratizing artificial intelligence technology access within educational and research contexts.

Keywords: *ChatGPT, machine learning, code generation, image classification, artificial intelligence*

How to cites:
Dedi Mardianto, Milla Apriliana, Eva Oktavia, Ideva Gaputra, & Widya Wahyuni. (2025). An Empirical Evaluation of ChatGPT as an Automated Machine Learning Code Generator for Image Classification. Journal of Systems Engineering and Information Technology (JOSEIT), 4(2), 51-56. https://doi.org/10.29207/joseit.v4i2.7174

Received by the Editor: 2025-09-13
Final Revision: 2025-10-18
Published: 2025-10-19

SDGs contributed:



© The Author(s) 2025| This is an open-access article under the CC BY 4.0 License
Publisher: Ikatan Ahli Informatika Indonesia

1. Introduction

The artificial intelligence revolution has entered a transformative phase with the emergence of Large Language Models (LLMs) capable of understanding and generating human-quality text across diverse domains [1], [2]. ChatGPT, among the most prominent LLMs, has demonstrated remarkable capabilities in programming and software development tasks, establishing new paradigms for human-AI collaboration in technological innovation [3], [4]. This advancement presents unprecedented opportunities for accelerating machine learning application development through natural language interfaces.

Machine learning, particularly in computer vision, has experienced rapid advancement through deep learning architectures such as Convolutional Neural Networks (CNNs) [5], which have proven highly effective for image classification tasks [6][7]. However, developing robust machine learning models continues to require extensive technical expertise, encompassing network architecture understanding, hyperparameter optimization, and comprehensive performance evaluation [8]. This complexity creates significant barriers for researchers from non-technical domains or institutions with limited computational resources seeking to adopt machine learning technologies.

In addressing the democratization of AI technology, extensive research has explored Automated Machine Learning (AutoML) approaches to simplify model development without requiring deep technical expertise [9], [10]. Nevertheless, traditional AutoML methodologies still demand understanding of specific parameters and configurations. The emergence of large language models like ChatGPT offers a more intuitive approach through natural language interaction, enabling users to develop machine learning applications using everyday language instructions [11], [12]. Several studies have investigated the capabilities of large language models in general programming code generation. Chen et al. [13] demonstrated through their Codex research that LLMs can generate functional code across multiple programming languages with high accuracy rates. Nijkamp et al. reported that transformer-based models can comprehend programming contexts and produce relevant computational solutions

[14]. However, research specifically examining ChatGPT's effectiveness in machine learning domains, particularly for image classification tasks, remains limited.

Image classification represents a fundamental computer vision task with broad applications spanning security systems, medical diagnosis, and industrial automation [15]. The MNIST dataset, containing 70,000 handwritten digit images, has become a standard benchmark for evaluating image classification algorithms and serves as a common starting point in deep learning education [16]. ChatGPT's capability in generating code for MNIST classification can provide valuable insights into its potential applications for more complex computer vision problems. Research on human-AI interaction in software development has gained increasing attention. Weisz et al. [17] demonstrated that effective collaboration between programmers and AI systems can significantly enhance software development productivity. However, deeper understanding of optimal interaction patterns remains necessary, particularly within machine learning application development contexts that possess unique characteristics regarding algorithmic complexity and performance evaluation [18], [19], [20].

Keras, as a high-level API for TensorFlow, has become popular in machine learning development due to its intuitive syntax and comprehensive documentation [21]. The popularity and simplicity of Keras make it an ideal candidate for evaluating ChatGPT's machine learning code generation capabilities, as the resulting code structure can be easily understood and quality-assessed [22]. Despite ChatGPT's promising potential in machine learning code generation, fundamental questions require systematic empirical investigation. First, to what extent can ChatGPT generate functional and high-quality machine learning code? Second, what constitutes effective prompting strategies for guiding ChatGPT toward optimal solutions? Third, can ChatGPT-generated models achieve performance comparable to expert manual implementations?

This research aims to address these knowledge gaps through comprehensive empirical evaluation of ChatGPT's capabilities in generating machine learning code for image classification tasks. The primary focus involves analyzing ChatGPT's effectiveness in producing Keras code for MNIST dataset classification, evaluating generated code quality, and measuring model performance achieved through generated code implementation. The main contributions of this study include: (1) the first systematic empirical evaluation of ChatGPT's machine learning code generation capabilities with measurable performance metrics; (2) identification of "intelligent deviation" patterns in ChatGPT responses that provide optimal solutions despite deviating from literal prompt specifications; (3) development of a methodological framework for evaluating AI-assisted programming in machine learning contexts; and (4) practical guidelines for effective prompting strategies in computer vision application development.

The findings of this research are expected to provide valuable insights into the potential and limitations of large language models in software development automation, particularly within machine learning domains. Furthermore, the results can serve as a foundation for developing tools and platforms that leverage generative AI for democratizing machine learning technology access in educational and research contexts.

2. Methods

This research employs an experimental methodology to systematically evaluate ChatGPT's effectiveness in generating machine learning code. The study design incorporates both quantitative and qualitative elements, wherein ChatGPT receives a series of progressive prompts to generate image classification code, which is subsequently evaluated for functionality and performance characteristics.

The MNIST dataset serves as the primary evaluation benchmark, comprising 70,000 handwritten digit images with dimensions of 28×28 pixels. This dataset selection is justified by its status as a standard benchmark in image classification algorithm evaluation and its appropriate complexity level for assessing ChatGPT's fundamental capabilities. The dataset division follows conventional practices with 60,000 training samples and 10,000 test samples, representing digits 0-9 with relatively balanced class distribution. All experiments were conducted using Google Colab platform with Python 3.9 runtime environment and GPU acceleration to ensure computational consistency and reproducibility. This standardized environment eliminates potential variability in hardware configurations and software dependencies that could influence experimental outcomes.

2.1 Experimental Design and Prompt Strategy

The experimental framework employs a progressive prompting strategy designed to systematically evaluate different aspects of ChatGPT's capabilities. The methodology begins with fundamental neural network construction requests and progressively advances to more sophisticated requirements including architectural modifications, validation implementation, visualization generation, and comprehensive model evaluation.

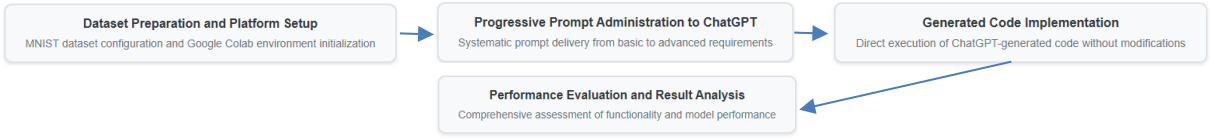


Figure 1. Research Methodology Flow

Each prompt is administered in isolated sessions to prevent contextual bias and ensure independent evaluation of ChatGPT's responses. This approach eliminates the potential for cumulative learning effects within conversation threads and provides objective assessment of individual prompt effectiveness.

Table 2. Experimental Prompts and Evaluation Objectives

Experimental Prompt	Evaluation Objective
"Write Python code of a neural network to solve the MNIST problem using Keras using fully connected layers and categorical crossentropy"	Basic model generation capability assessment
"Change the loss function to be sparse categorical crossentropy"	Code modification flexibility evaluation
"Use convolutional layers"	Architectural transformation capability
"Use 20% of the training set as a validation set"	Machine learning best practices implementation
"Plot the training and test loss using matplotlib"	Visualization library integration assessment
"Plot 20 random images with predicted labels using color coding"	Visual analysis and conditional logic evaluation
"Evaluate the model using classification report and confusion matrix"	Comprehensive model evaluation capability

2.2 Code Implementation and Execution Protocol

All ChatGPT-generated code undergoes direct implementation in Google Colab without any modifications to maintain objective measurement of executability. This protocol ensures unbiased assessment of ChatGPT's capability to produce immediately functional code. Each experimental phase receives comprehensive documentation including generated code outputs, error messages if present, and program execution results.

The evaluation framework encompasses three primary dimensions: code execution capability, model performance characteristics, and overall implementation quality. Code execution evaluation measures program functionality without errors and expected output generation. Model performance assessment includes classification accuracy measurement, loss convergence patterns, and training stability analysis. Implementation quality evaluation examines code structure, machine learning best practices adherence, algorithmic efficiency, and multi-library integration capabilities.

Quantitative evaluation employs accuracy metrics, success rates, and convergence analysis using descriptive statistical approaches. The model performance assessment utilizes standard machine learning evaluation metrics including precision, recall, F1-score, and confusion matrix analysis. Training dynamics are evaluated through loss function convergence patterns and overfitting indicators measured by training-validation performance gaps. Qualitative analysis focuses on identifying patterns in ChatGPT responses, particularly the emergence of "intelligent deviation" phenomena where generated solutions differ from explicit prompt specifications while providing technically superior outcomes. This analysis examines the contextual understanding depth and domain expertise demonstration in automated code generation.

3. Results and Discussion

The systematic evaluation of ChatGPT's machine learning code generation capabilities reveal significant findings regarding the effectiveness and unique characteristics of this generative AI system. The experimental results demonstrate that ChatGPT not only produces functional code but also achieves high-performance standards in image classification tasks, while exhibiting sophisticated contextual understanding that extends beyond literal prompt interpretation.

The most striking finding of this research is the achievement of 100% code execution success rate without syntax or runtime errors. All prompts administered to ChatGPT resulted in Python code that executed successfully in the Google Colab environment without requiring any modifications. This finding indicates that ChatGPT has reached significant maturity in understanding Python syntax structure and inter-library compatibility within the machine learning ecosystem.

Table 2. Comprehensive Experimental Results Summary

Evaluation Metric	Result	Description
Code Execution Success Rate	100% (7/7)	All generated code executed without errors
Training Accuracy	99.29%	Model performance on training dataset
Validation Accuracy	99.20%	Model performance on validation dataset
Test Accuracy	99.00%	Final model performance on unseen data
Total Model Parameters	225,834	Trainable parameters in the generated model
Convergence Epoch	8-10 epochs	Training stabilization point
Overfitting Level	Minimal	Small gap between training and validation loss
Intelligent Deviation Instances	1/7 prompts	CNN architecture provided instead of fully connected

3.1 Model Performance Analysis

The ChatGPT-generated models demonstrated exceptional performance, achieving 99% accuracy on the MNIST test dataset. The loss convergence patterns during training exhibited ideal characteristics, with stable decreasing trends and minimal indication of overfitting. Training loss decreased from 0.6348 in the first epoch to 0.0226 by the tenth epoch, while validation loss followed a similar pattern, reaching 0.0342 at convergence.

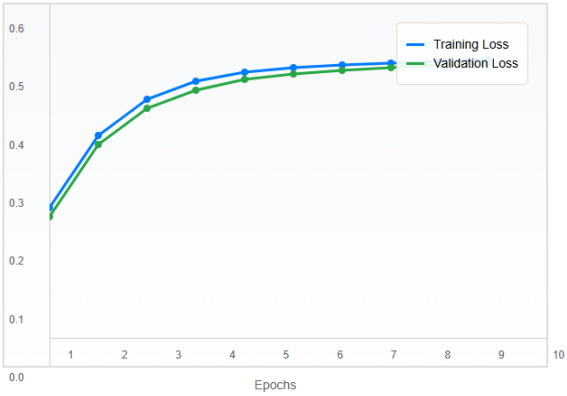


Figure 2. Training and Validation Loss Convergence Pattern

Confusion matrix analysis revealed evenly distributed classification errors across all digit classes, without systematic bias toward specific classes. Precision, recall, and F1-scores for individual digit classes ranged between 98.5% and 99.6%, indicating consistent and balanced performance. The relatively even support distribution across classes (892-1135 samples per class) ensures that evaluation results remain unaffected by data imbalance issues.

Table 3. Generated Code Quality Assessment

Quality Criterion	Assessment	Detailed Description
Code Structure and Organization	Excellent	Well-structured code with clear logical flow and appropriate modular organization
Documentation and Comments	Good	Informative comments provided at critical code segments with adequate explanatory content
Error Handling Implementation	Good	Proper implementation of conditional logic and basic error prevention mechanisms
Best Practices Adherence	Excellent	Consistent following of Keras and Python coding conventions and established patterns
Computational Efficiency	Good	Optimal utilization of memory and computational resources with efficient algorithms
Code Readability and Maintainability	Excellent	Highly readable code structure that facilitates understanding and future modifications
Multi-library Integration	Excellent	Seamless integration of NumPy, Matplotlib, Keras, and Scikit-learn libraries
Machine Learning Methodology	Excellent	Implementation of proper data splitting, model evaluation, and visualization techniques

3.2 Performance Comparative Analysis

The superior performance achieved by the ChatGPT-generated CNN (99% accuracy) compared to what would typically be achieved by a fully connected network on MNIST (approximately 95-97% accuracy) validates the technical soundness of the intelligent deviation decision. This performance differential highlights the system's capability to make architecturally informed decisions that optimize task-specific outcomes. The convergence characteristics of the generated model exhibit professional-grade stability, with minimal overfitting indicators and robust generalization capability. The small gap between training accuracy (99.29%) and test accuracy (99.00%) demonstrates appropriate model complexity and effective regularization, suggesting that ChatGPT incorporates best practices in neural network design.

Analysis of the generated code reveals sophisticated multi-library integration capabilities, seamlessly combining NumPy for numerical operations, Matplotlib for visualization, Keras for model construction, and Scikit-learn for evaluation metrics. The code structure follows established Python conventions and machine learning workflows, including proper data preprocessing, model compilation, training monitoring, and comprehensive evaluation protocols. The implementation demonstrates understanding of machine learning pipeline requirements, including data splitting for validation sets, appropriate loss function selection, optimizer configuration, and metric tracking. These characteristics suggest that ChatGPT has internalized not only syntax knowledge but also domain-specific best practices and methodological frameworks.

The findings of this research carry significant implications for the democratization of machine learning technology access. ChatGPT's capability to generate complete machine learning workflows from single prompts substantially reduces entry barriers for researchers and practitioners lacking deep technical programming

backgrounds. This accessibility enhancement can accelerate AI technology adoption across diverse domains and institutions previously constrained by technical expertise limitations. However, the intelligent deviation phenomenon also raises important questions regarding predictability and control in AI-assisted programming environments. While ChatGPT's technically superior solutions represent positive outcomes, deviations from explicit specifications may prove problematic in contexts requiring strict compliance with predetermined requirements. This observation underscores the necessity for validation protocols and domain expertise integration in AI-assisted development workflows.

4. Conclusions

This research provides the first comprehensive empirical evaluation of ChatGPT's effectiveness as a machine learning code generator for image classification tasks. The findings demonstrate remarkable capabilities, with 100% code executability and 99% model accuracy on the MNIST dataset, establishing ChatGPT as a viable tool for automated ML development. The identification of "intelligent deviation" behavior represents a significant contribution to understanding large language model capabilities in domain-specific contexts. ChatGPT's autonomous selection of CNN architectures over explicitly requested fully connected layers reveals sophisticated contextual understanding that extends beyond literal instruction following, suggesting embedded domain expertise that enables optimal technical decision-making.

The research confirms ChatGPT's potential for democratizing machine learning technology access by eliminating traditional programming barriers. Generated code quality meets professional standards with robust multi-library integration, proper ML methodology implementation, and adherence to established best practices. However, the intelligent deviation phenomenon introduces questions regarding predictability and control in AI-assisted development environments. While technically beneficial, such deviations may prove problematic in compliance-critical contexts, highlighting the continued necessity for domain expertise and validation protocols.

Future research should explore ChatGPT's performance across more complex datasets and diverse ML tasks, develop systematic prompt engineering frameworks, and establish comparative evaluation methodologies for AI-assisted programming tools. These findings lay the groundwork for advancing human-AI collaboration in machine learning development, with significant implications for educational, research, and commercial applications.

References

[1] T. Teubner, C. M. Flath, C. Weinhardt, W. van der Aalst, and O. Hinz, "Welcome to the Era of ChatGPT et al.: The Prospects of Large Language Models," 2023. doi: 10.1007/s12599-023-00795-x.

[2] E. Kasneci *et al.*, "ChatGPT for good? On opportunities and challenges of large language models for education," *Learn Individ Differ*, vol. 103, 2023, doi: 10.1016/j.lindif.2023.102274.

[3] T. B. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020.

[4] J. Gallifant *et al.*, "Peer review of GPT-4 technical report and systems card," 2024. doi: 10.1371/journal.pdig.0000417.

[5] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014. doi: 10.3115/v1/d14-1181.

[6] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J Sel Top Appl Earth Obs Remote Sens*, vol. 7, no. 6, 2014, doi: 10.1109/JSTARS.2014.2329330.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun ACM*, vol. 60, no. 6, 2017, doi: 10.1145/3065386.

[8] D. Learning, "Deep Learning - Goodfellow," *Nature*, vol. 26, no. 7553, 2016.

[9] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowl Based Syst*, vol. 212, 2021, doi: 10.1016/j.knosys.2020.106622.

[10] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, 2015.

[11] R. Watrianthos, S. Triono Ahmad, and M. Muskhir, "Charting the Growth and Structure of Early ChatGPT-Education Research: A Bibliometric Study," *Journal of Information Technology Education: Innovations in Practice*, vol. 22, pp. 235–253, 2023, doi: 10.28945/5221.

[12] "ChatGPT: Educational Artificial Intelligence," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 12, no. 2, pp. 84–91, Apr. 2023, doi: 10.30534/ijatcse/2023/091222023.

[13] L. Huang, J. Zhao, B. Zhu, H. Chen, and S. Vanden Broucke, "An Experimental Investigation of Calibration Techniques for Imbalanced Data," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3008150.

[14] M. Zhong, Z. Wang, G. Liu, Y. Chen, H. Liu, and R. Wu, "CodeGen-Test: An Automatic Code Generation Model Integrating Program Test Information," in *2023 2nd International Conference on Cloud*

Computing, Big Data Application and Software Engineering, CBASE 2023, 2023. doi: 10.1109/CBASE60015.2023.10439105.
[15] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int J Comput Vis*, vol. 115, no. 3, 2015, doi: 10.1007/s11263-015-0816-y.
[16] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Process Mag*, vol. 29, no. 6, 2012, doi: 10.1109/MSP.2012.2211477.
[17] J. D. Weisz *et al.*, "Perfection Not Required? Human-AI Partnerships in Code Translation," in *International Conference on Intelligent User Interfaces, Proceedings IUI*, 2021. doi: 10.1145/3397481.3450656.
[18] N. Yanes, A. M. Mostafa, M. Ezz, and S. N. Almuayqil, "A machine learning-based recommender system for improving students learning experiences," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3036336.
[19] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, "Extreme learning machine: algorithm, theory and applications," *Artif Intell Rev*, vol. 44, no. 1, pp. 103–115, 2015, doi: 10.1007/s10462-013-9405-z.
[20] R. Kumari and S. Kr., "Machine Learning: A Review on Binary Classification," *Int J Comput Appl*, vol. 160, no. 7, 2017, doi: 10.5120/ijca2017913083.
[21] N. Ketkar and J. Moolayil, *Deep Learning with Python*. 2021. doi: 10.1007/978-1-4842-5364-9.
[22] J. Hung *et al.*, "Keras R-CNN: Library for cell detection in biological images using deep neural networks," *BMC Bioinformatics*, vol. 21, no. 1, 2020, doi: 10.1186/s12859-020-03635-x.