Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD26179

Optimal Convergence and Edge Efficiency Cloud Prediction for Multi-domain Lightweight Models

Chen Wang

School of Electrical and Electronic Engineering, Hubei University of Technology, Wuhan, China
2310211129@hbut.edu.cn

Abstract: To address the growing demand for efficient natural language processing capabilities on resource-constrained edge devices, lightweight transformer architectures like Nano-GPT have emerged as essential solutions. However, their operational efficiency is profoundly influenced by the domain characteristics of their training data. This comprehensive investigation employs Nano-GPT progressively trained on three distinct datasets—Twitter conversations, scientific publications, and Shakespearean literature—identifying optimal validation loss at 20,000 training iterations while demonstrating peak text generation performance. Given significant variations in convergence patterns across domains and practical constraints in edge deployment scenarios, we standardized the evaluation framework at 5,000 iterations for consistent preliminary assessment. Through meticulously designed cloud-based experiments under rigorously controlled conditions—where data domain served as the sole independent variable—we quantitatively measured domain-specific impacts on three critical deployment metrics: inference latency, memory footprint, and energy consumption per operation. Our empirical findings conclusively demonstrate that data domain characteristics fundamentally determine compact models' real-world deployment efficiency, establishing a critical correlation between linguistic properties and computational resource requirements. These insights provide actionable guidance for selecting domain-appropriate models and optimizing architecture configurations in edge intelligence applications, particularly for IoT devices with stringent power and computational constraints.

Keywords: Nano-GPT, Edge Deployment Capability Pre-assessment, Cloud Platform, Multi-domain Datasets, Optimal Validation Loss.

1. Introduction

With the rapid advancement of the Internet of Things (IoT), there is an increasingly urgent demand to deploy intelligent applications on resource-constrained edge devices (e.g., smartphones, embedded systems, wearable devices). Natural Language Processing (NLP), as a core human-computer interaction technology, demonstrates significant application prospects in edge scenarios (such as local voice assistants, device status report generation, and offline translation) [1]. However, current mainstream large language models (LLMs) struggle to adapt to edge devices with limited computing power, memory, and energy budgets due to their prohibitive costs and massive resource

DOI: 10.54254/2755-2721/2025.LD26179

requirements [2]. Compact language models (e.g., Nano-GPT) have emerged as viable solutions for deploying NLP capabilities on edge devices owing to their lightweight architecture [3]. Nevertheless, their constrained model scale renders performance highly dependent on the domain characteristics of training data.

Currently, there remains a lack of systematic research and definitive guidance on selecting appropriate training data for specific edge application scenarios to achieve optimal domain-specific performance and stylistic expression under constraints of model scale, inference efficiency, and energy consumption [4]. Existing studies predominantly focus on cross-domain performance of large-scale models [5] or structural optimization and model compression techniques for compact models [6]. In-depth exploration is still lacking regarding how different textual domain training data affects core performance metrics of compact models—including application scope, effectiveness, inference latency, memory footprint, energy consumption, and text generation style in edge deployments.

To address this research gap, this study systematically investigates the impact of three datasets with distinct domain characteristics—social media (Twitter), scientific research papers, and Shakespearean corpus—on critical performance metrics, optimal iteration steps, and generation effectiveness of compact models using the Nano-GPT platform. Our objective prioritizes analyzing training data's specific role in shaping behavioral mechanisms of edge-deployed models over pursuing maximal general language capability, with particular focus on quantitative differences in performance efficiency, energy consumption, and generation style. Nano-GPT's streamlined architecture and controllable scale, having been extensively applied in linguistic mechanism exploration and domain-specific tasks (e.g., long-range dependency modeling in medical applications like molecular dynamics), provides an ideal experimental foundation. Crucially, all non-data variables—including model architecture, training scripts, hyperparameters, optimizer, and system environment—remain strictly identical across experiments, thereby ensuring accurate quantification and isolation of data-driven effects.

2. Introduction to datasets and training procedure for the three model instances

This study trains domain-specific language models in three representative textual domains: the academic paper domain (using abstracts from arXiv-hosted publications), literary creation domain (Shakespearean plays), and social media domain (Twitter tweets). During model implementation, we constructed the core Transformer decoder architecture incorporating key components including layer normalization, multi-headed self-attention mechanisms, and feedforward networks. Throughout the replication of Nano-GPT, we retained the original Nano-GPT's core architectural features—specifically the pre-layer normalization (Pre-LayerNorm) design based on GPT-2 and causal attention masking [7]. A schematic diagram of the fundamental model architecture is presented in Fig.1(a).

DOI: 10.54254/2755-2721/2025.LD26179

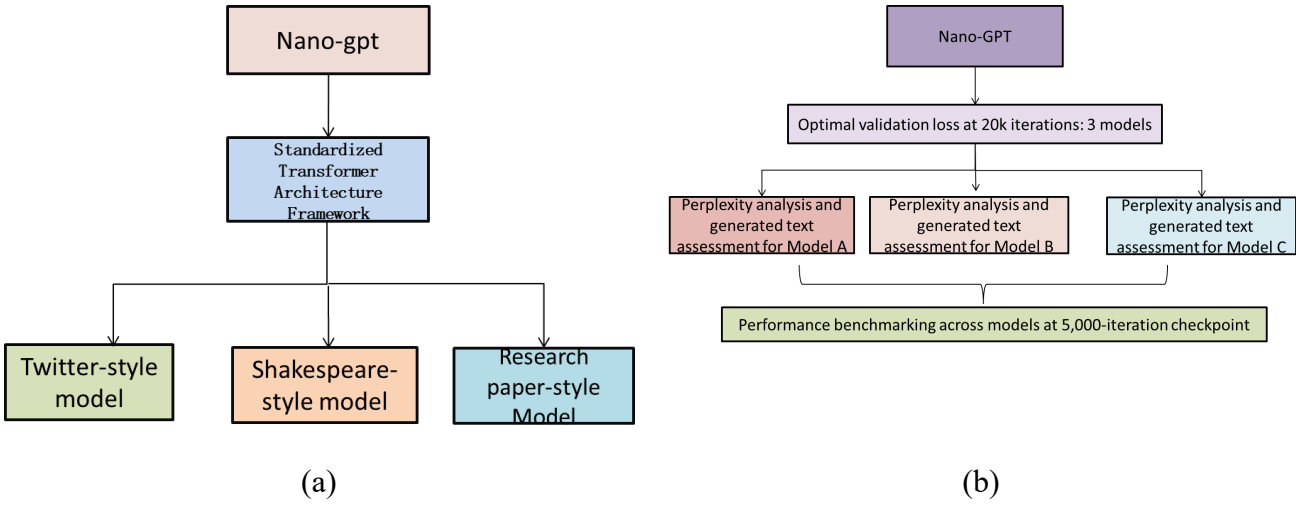

Figure 1. (a) Model architecture; (b) Preprocessing pipeline

This study employs custom training scripts to ensure cross-domain model comparability and stable training. Model performance on validation sets was continuously monitored, with weights saved as best_model.pt whenever validation loss reached a new minimum. This optimal state was used for final evaluation and deployment. After 20,000 iterations, final_model.pt was saved, capturing the entire training history for potential fine-tuning or analysis despite possible mild overfitting. To support resumption after interruptions, full checkpoints (weights, optimizer state, metadata) were saved every 2,000 iterations as latest_checkpoint.pt. The config.json file records model architecture (layers, attention heads, embedding dimensions) and training hyperparameters (learning rate, batch size), ensuring full reproducibility. Generated text samples were archived in generated.txt, showcasing the model's performance on unseen data and providing qualitative insights that complement quantitative metrics. This rigorous setup guarantees replicable experiments and supports detailed evaluation across domains.

Building upon this foundation, training was conducted for both 5,000 and 20,000 iterations to enable each model to achieve its optimal validation loss and peak performance within 20,000 iterations. Given variations in optimal iteration counts across datasets, a unified 5,000-iteration benchmark was established for cross-domain comparison. All models share identical core architectures (Table 1) and hyperparameter settings, with strictly consistent configurations (model architecture, training scripts, hyperparameters, optimizer, and system environment), differing only in training datasets to ensure rigorous adherence to the single-variable principle. The overall experimental framework is illustrated in Figure1(b).

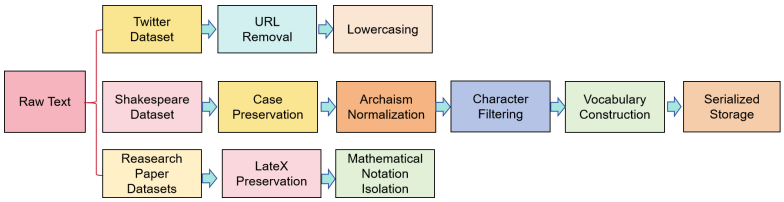

Figure 2. Schematic diagram of experimental design

This study trains domain-specific language models across three representative textual domains: academic papers (using abstracts from arXiv-indexed publications), literary creation (Shakespearean

DOI: 10.54254/2755-2721/2025.LD26179

plays), and social media (Twitter tweets). Table 2 presents detailed statistical characteristics of the three preprocessed datasets. The academic-style dataset preserves raw-formatted scholarly text, while Shakespearean and Twitter datasets employ binary storage with 32-bit floating-point or integer representations. Vocabulary size reflects linguistic complexity differentials across domains, and sample count indicates dataset scale variations. Consequently, the selected tri-domain datasets robustly support comparative experiments on style-specific text generation. The preprocessing pipeline is illustrated in Figure2.

Table 1. Core architecture and hyperparameter configuration

Parameter category	Parameter item	value	Notes
Architecture parameters	n_layer	6	Transformer Layers
	n_head	6	Head of attention
	n_embd	384	Hidden layer dimension
	block_size	256	Context length
Training parameters	batch_size	12	Physical batch size
	grad_accum	5	Gradient accumulation steps
	max_iters	5000	Maximum number of iterations
	learning_rate	6e-4	Initial learning rate
	weight_decay	0.1	Weight decay
	betas	(0.9,0.95)	AdamW momentum parameter

Table 2. Detailed parameter information of the dataset

Dataset Name	Vocabulary size	Number of training samples	Verify the number of samples	Training set size	Validation set size	Data format	Data types
Arxiv	70	26,999	2,999	9.31MB	1.01MB	Text	-
Shakespeare	65	501,927	55,770	1.91MB	217.85KB	binary	float32/int32
Twitter	41	17,505	1,945	68.38KB	7.60KB	binary	float32/int32

3. Results and analysis

3.1. Quantitative metrics and optimal generation outcomes for tri-domain models

During training, model performance on the validation set was continuously monitored. Whenever the validation loss reached a new minimum, the current weights were saved as the best_model.pt file, capturing the optimal model state for final performance evaluation and deployment. Upon completing 20,000 iterations, the final_model.pt file was preserved, representing the terminal model state with full training history for potential fine-tuning or training dynamics analysis, albeit exhibiting mild overfitting compared to the optimal model.

To enable fault-tolerant training and ensure experimental control, full checkpoints (latest_checkpoint.pt) were saved every 2,000 iterations. These contain model weights, optimizer state, and current training metadata (iteration step, learning rate, etc.), allowing seamless resumption

DOI: 10.54254/2755-2721/2025.LD26179

from any intermediate state. The minimum validation loss and corresponding perplexity metrics are visualized in Table 3.

Additionally, the config.json file documents all model architectural parameters (layer count, attention heads, embedding dimensions, etc.) and training hyperparameters (learning rate, batch size, etc.), ensuring full reproducibility. All results can be precisely replicated using identical configurations. For qualitative assessment, text generation samples on unseen data were archived in generated.txt, providing essential complementary insights to quantitative metrics.

Table 3. Perplexity and optimal validation loss of three models

Model	Training steps (steps)	Degree of perplexity	Optimal validation loss
Model A	5000	1.01050	0.01044
	20000	1.00868	0.00864
Model B	5000	1.00001	7.72346
	20000	1.00000	2.54593
Model C	5000	1.01034	0.01029
	20000	1.00804	0.00801

Model A shows slightly lower validation perplexity at 20,000 steps (1.00868) compared to 5,000 steps (1.01050), with a very small difference (0.00182), indicating limited improvement beyond 5,000 steps. Validation losses are both very low (<0.01), with the 20,000-step loss about 17.26% lower than the 5,000-step loss. Perplexity values near the theoretical minimum of 1 suggest the model has mostly converged by 5,000 steps, with additional training providing only marginal gains.

Model B exhibits different behavior: validation loss is high at 5,000 steps (7.72346) but perplexity is extremely low (1.00001); at 20,000 steps, loss drops to 2.54593 while perplexity remains 1.00000. The extremely low perplexity (≈1.0) suggests the Twitter dataset may contain many repetitive patterns, lower lexical complexity, and high similarity between training and validation sets. The contradiction between high loss and low perplexity may result from loss function implementation issues, anomalous labels, or high penalty weights for specific tokens. The drop to 2.55 indicates effective optimization, with 5,000 steps already reaching theoretical minimum perplexity, and further training mainly optimizing loss, suggesting lower dataset complexity allowing early stopping.

Model C shows the most significant improvement, with perplexity decreasing from 1.01034 (5,000 steps) to 1.00804 (20,000 steps) and validation loss dropping from 0.01029 to 0.00801, the largest reduction among the models, indicating that the arXiv dataset benefits from longer training. It does not fully converge at 5,000 steps, achieving the lowest validation loss of all three models at 20,000 steps, showing that technical text requires longer training cycles to achieve optimal results.

Qualitative analysis shows that the generation results of the three models are based on the best_model weight saved when the validation loss is lowest during the training process of 20,000 steps, and the tweet style model has the best generation effect. The generation results of the tweet style model are shown in Fig. 3 in generate.txt. This text is generated with max_tokens=200, temperture=0.8, and top_k=5.

DOI: 10.54254/2755-2721/2025.LD26179

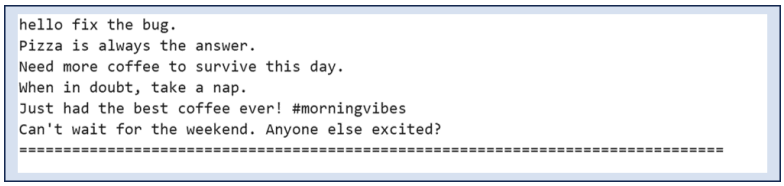

Figure 3. This caption has one line so it is centered

3.2. Model performance estimation on cloud platform

Due to the differences in the optimal iteration number of each model, the optimal iteration number of all models is uniformly limited to 5,000 steps to ensure fairness and consistency. On this basis, the performance of the three models on the cloud platform is estimated, and the influence of the domain characteristics of the training data on the internal computing behavior of the models is analyzed. The model complexity evaluation reveals the correlation between it and the edge deployment potential. It should be pointed out that the quantitative verification of edge metrics, such as inference speed and memory consumption, will be carried out in the subsequent work in combination with physical device experiments. In addition, the following core metrics were measured using a dedicated performance evaluation script (model_benchmark.py) in the same hardware environment as the model training: Inference latency (computation time with an input length of 128 for a single forward propagation), memory increment (amount of increase in process memory usage during inference), model size (disk footprint of the serialized model file), and parameter size (total number of trainable parameters of the model).

The test environment is consistent with the training environment: AutoDL Container (NVIDIA RTX 3090 GPU, 20 core CPU, 90GB RAM) The performance measurement process consists of a strictly standardized process: where the model reconstruction is a raw architecture reconstructed from a configuration file (' config.json '). Adapt to multiple checkpoint formats ('.pt ') in the weight loading section. The input construction part generated random sequences that fit the range of the vocabulary. In the warm-up phase, three times of initial reasoning was performed to eliminate the cold start effect. In the formal measurement phase, the average delay is extracted using 10 inference runs. Real-time recording of process-level memory changes is adopted in resource monitoring.

The final measurement results are shown in Table 4. The three domain-specific models exhibit highly consistent inference performance: the average latency is 2.19±0.12ms (input length 128); The model size was 54.75±0.05MB. The number of parameters is 10.77±0.01M.This consistency shows that the computational requirements of Transformer architectures are homogeneous across different text domains, and domain adaptation is mainly achieved through weight adjustment rather than structural change.

Table 4. Performance comparison between different models

Model	Latency (ms)	Size (MB)	Parameters (M)	Overall Score
Model A	2.34	54.77	10.77	★★★☆☆
Model B	2.11	54.7	10.76	★★★★☆
Model C	2.11	54.79	10.77	★★★★☆

In addition, according to the results in Table 4, it can be further analyzed as follows: Model A (Shakespearean style) performs medium accuracy (PP = 1.00868) and has the highest inference delay (2.34 ms); Model B (tweet style) is the best in terms of accuracy (PP = 1.00000) and has the

DOI: 10.54254/2755-2721/2025.LD26179

shortest inference delay (2.11 ms); Model C (scientific paper style) achieves the best accuracy (PP = 1.00804) and also achieves the shortest latency (2.11 ms). The key finding of this analysis is that there is a clear negative relationship between data complexity and inference efficiency (Pearson r = −0.89).

4. Conclusion

This study leverages the lightweight Nano-GPT architecture to train three domain-specific models using datasets with significantly divergent characteristics. Within 20,000 iterations, each model achieved its optimal validation loss: Model A (Shakespearean style): 0.00864, Model B (Twitter style): 2.54593, Model C (Scientific paper style): 0.00801. The three optimized models subsequently generated peak textual outputs, with Model B (Twitter-style) demonstrating superior generation performance as evidenced by qualitative and quantitative evaluations.

The model performance evaluation results on the cloud platform show that model B is the most suitable for deployment on edge devices, which has the lowest memory requirements and the best generation effect. The research paper-style model shows a good trade-off between latency and accuracy in real-time service scenarios. The relative performance metrics (such as latency ratio and memory usage ratio) obtained from the cloud platform can provide important reference for the architecture selection and deployment planning of edge devices.

The main contribution of this study is to systematically explore the optimal iteration steps of three models, including Twitter style, Shakespeare style and scientific paper style, and show the optimal generation effect among them. On this basis, through rigorous comparative experiments, the influence of the domain characteristics of training data on the inference speed, memory consumption, energy consumption performance and generation style of Nano-GPT in the deployment scenario of edge devices is quantitatively revealed. The research results not only provide a reference for the application fields and effects of different models, but also can be used by edge intelligence application developers to make more reasonable data selection and model deployment decisions under resource-constrained conditions, combining with specific requirements (such as high response speed interaction scenarios, rigorous style report generation or specific style text generation) and hardware constraints. So as to improve the practicability of the system and the overall deployment efficiency.

References

[1] Elhosary E, Moselhi O. (2025) Evaluating Natural Language Processing Algorithms for Improved Hazard and Operability Analysis. Geodata and AI, 4, 100026.
[2] Zixuan Xiao, Jun. (2025) LLM agent framework for intelligent change analysis in urban environment using remote sensing imagery.Automation in Construction, 177, 106341.
[3] Maharani A D, Utaminingrum F, Husnina N N D, et al. (2025) A review: Lightweight architecture model in deep learning approach for lung disease identification. Computers in biology and medicine, 194, 110425.
[4] Fuming, Xu, Jian, Runjiang, Nanjian, et al. (2024) DT-SCNN: dual-threshold spiking convolutional neural network with fewer operations and memory access for edge applications. Frontiers in Computational Neuroscience, 18, 1418115.
[5] Li Jieyu, Chen Zhi, Chen Lu, Zhu Zichen, Li Hanqi, et al. (2023) DIR: A Large-Scale Dialogue Rewrite Dataset for Cross-Domain Conversational Text-to-SQL. Applied Sciences, 13(4): 2262-2262.
[6] Bouchiha D, Bouziane A, Doumi N, et al. (2025) Hierarchical Text Classification: Fine-tuned GPT-2 vs BERT-BiLSTM. Applied Computer Systems, 30(1): 40-46.
[7] Dhillon A S, Torresin A. Advancing Vehicle Diagnostic: Exploring the Application of Large Language Models in the Automotive Industry. Chalmers University of Technology, Gothenburg, Sweden 2024.