An Empirical Investigation into the Limitations of Sparse Mixture of Experts for Small Scale Character Level Modeling

Fuguo Xu, Wangwang Hou, and Liangxiong Wei

***Abstract*—Mixture-of-Experts (MoE) models have demonstrated impressive scaling capabilities for large language models, yet their efficacy in small-scale, character-level language modeling remains unclear. We conduct a systematic study to evaluate whether sparse expert routing can improve generalization and efficiency under limited computational budgets. Adapting MoE to this setting is challenging due to auxiliary load-balancing losses that can destabilize training, increased parameter counts that may cause overfitting, and routing overhead that could negate theoretical gains. To address this, we implement a Switch-Transformer-style MoE layer with top-1 routing and an auxiliary load-balancing loss, comparing multiple MoE variants (varying the number of experts, auxiliary loss coefficient, and expert dropout) across three character-level datasets (shakespeare_char, enwik8, text8). Our results show that all MoE configurations increase parameter count and training time, yet yield higher validation loss and slower inference speed compared to dense baselines. For instance, on shakespeare_char the baseline achieves a best validation loss of 1.46 bits per character, while the 4-expert MoE reaches 1.53 bits per character. We also find that stronger auxiliary loss coefficients (e.g., 0.1 vs 0.01) destabilize training, and expert dropout fails to improve generalization. These findings indicate that the overhead of sparse routing outweighs its potential benefits in small-scale character-level modeling, providing valuable insights for future work on efficient expert utilization. The key contributions of this study include: (1) isolating the sparse routing effect in a character-level, low-resource regime; (2) providing a mechanistic explanation of routing failure via gradient conflict, expert specialization entropy, and computational overhead analysis; (3) establishing clear scale boundaries that delineate when MoE becomes detrimental versus beneficial.**

***Key Words*—Character-level language modeling, Mixture-of-Experts, Sparse expert routing, Transformer, Small-scale models**

I. Introduction

TRANSFORMER architectures [1] have become the de facto standard for modern language modeling, enabling breakthroughs in natural language understanding and generation. Scaling these models to larger parameter counts typically yields improved performance, but at immense computational cost[2, 3]. Mixture-of-Experts (MoE) models address this by activating only a subset of expert sub-networks per token, increasing capacity while keeping per-token FLOPs low[4]. While MoE has demonstrated impressive scaling for large language models, its effectiveness in small-scale, character-level language modeling remains largely unexplored [5].

Relationship between sparse expert routing and character-level modeling.The fine-grained nature of character-level tokens introduces distinct challenges for sparse routing. Because each character carries less semantic information than a subword or word token, the router must base its decisions on weaker contextual signals[6]. Furthermore, the longer sequences typical of character-level data increase the frequency of routing decisions, amplifying any overhead and potentially magnifying instabilities[7]. Consequently, the benefits of sparsity observed in large-scale word-level models—where tokens are richly informative, and routing is relatively infrequent—may not translate directly to the character-level regime[8].

Specifically, character-level tokens have a much smaller vocabulary (typically a few hundred symbols) compared to subword vocabularies (tens of thousands). This reduces the entropy of the token distribution, making it harder for the router to differentiate between tokens and assign them to specialized experts[9]. Moreover, the contextual information needed for routing decisions is spread across longer sequences, demanding a broader receptive field—a requirement that may conflict with the limited context length used in small-scale models. The combination of weak semantic signals, high routing frequency, and low differentiation among tokens creates a "routing noise" effect, where experts receive noisy gradients that hinder specialization.

The relationship can be further understood through three intertwined aspects:

- **Token information density:** Each character carries minimal semantic content, forcing the router to rely on subtle contextual patterns that are often ambiguous. This low signal-to-noise ratio leads to high variance in routing decisions, preventing stable expert specialization.
- **Routing frequency:** Character-level sequences are typically 4–5 times longer than subword sequences for the same text span[10], resulting in a proportional increase in routing operations. The cumulative overhead of these frequent decisions can erase any theoretical FLOPs advantage, especially when the expert networks are relatively shallow.

Manuscript received January 19, 2026; revised January 26 and February 2, 2026; accepted March 13, 2026. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

This work was supported by the Doctoral Research Start-up Fund of Shangqiu Normal University under Grant No.SQNUQDF2553.

F. Xu, W. Hou, and L. Wei are with the College of Information Technology, Shangqiu Normal University, Shangqiu 476000, China (email: fuguoxu@ieee.org, houww_sqnu@163.com, weilx_scu@aliyun.com).

Corresponding Author: LiangXiong Wei.

- **Capacity mismatch:** Small-scale models have limited hidden dimensions (e.g., 384 in our baseline), which restricts the expressive power of each expert. When experts are too weak to develop distinct skills, the router cannot assign tokens meaningfully, and the MoE layer degenerates into an ensemble of slightly perturbed copies of a single FFN.

Investigating this relationship is crucial for determining whether sparse expert routing can be effective in resource-constrained settings that rely on character-level inputs, and for identifying the architectural modifications needed to make it work.

Thus, the interplay between sparse expert routing and character-level modeling is fundamentally shaped by the low information density per token, high routing frequency, and limited expert capacity inherent to small-scale settings. This interplay underpins our investigation and explains why the standard MoE approach fails to deliver benefits in this regime.

Character-level modeling presents a unique challenge due to longer sequences and a fine-grained symbol space, making it a common testbed for studying efficiency–generalization trade-offs under constrained compute budgets [11]. Understanding whether sparse expert routing can improve generalization and efficiency in this regime is relevant for resource-constrained applications and for probing the fundamental limits of sparse architectures.

However, adapting MoE to small-scale character-level transformers is non-trivial. The auxiliary load-balancing loss, essential for preventing expert collapse, can destabilize optimization[12]; the increased parameter count may lead to overfitting [13]; and the routing overhead may outweigh any theoretical gain from sparsity [14]. Moreover, the typical assumptions of large-scale MoE (e.g., abundant data, many experts) do not hold in our setting, making it unclear whether existing MoE techniques will transfer.

In this work, we conduct a systematic study of sparse expert routing in character-level transformers. We implement a Switch-Transformer-style MoE layer with top-1 routing and an auxiliary load-balancing loss, and evaluate it across three character-level datasets (shakespeare_char, enwik8, text8). Our experiments compare dense baselines against multiple MoE variants, varying the number of experts, auxiliary loss strength, and the use of expert dropout.

The main contributions of this work are as follows.

- **Implementation of a sparse MoE layer** for a character-level GPT architecture, supporting top-1 routing and an auxiliary load-balancing loss, fully integrated into the training loop.
- **Extensive empirical evaluation** on three datasets, measuring validation loss (bits per character), training time, and inference speed for dense baselines and multiple MoE variants.
- **Analysis of key hyper-parameters**, showing that increasing the number of experts (from 2 to 4) does not improve generalization, that an overly strong auxiliary loss coefficient destabilizes training, and that expert dropout fails to provide regularization benefits.
- **Insights into the limitations of sparse expert routing** for small-scale language modeling: although MoE increases capacity, it underperforms dense baselines in both validation loss and inference speed, indicating that the overhead of routing and the added optimization difficulty outweigh the potential advantages at this scale.

We verify our findings through rigorous experimentation, training all models from scratch with the same optimization setup (AdamW [15, 16], cosine learning rate decay, and gradient clipping) and repeating each run with multiple random seeds for statistical reliability. Our results show that all MoE configurations increase parameter count and training time, yet yield higher validation loss and slower inference speed compared to dense baselines. For example, on shakespeare_char the baseline achieves a best validation loss of 1.46 bits per character, while the 4-expert MoE reaches 1.53 bits per character, confirming that the benefits of sparsity observed in large-scale settings do not transfer directly to small-scale character-level tasks.

In the remainder of this paper, we review related work (Section II), provide problem formulation (Section III), detail our method (Section IV) and experimental setup (Section V), present results (Section VI), and discuss conclusions (Section VIII).

II. Related Work

Our study lies at the intersection of sparse expert routing and character-level language modeling. We compare and contrast prior work along three axes: Transformer scaling, Mixture-of-Experts (MoE) architectures, and character-level modeling.

A. Transformer Scaling

Transformers have been scaled to hundreds of billions of parameters [17, 18], with efficiency becoming a central concern. Techniques such as sparse attention [19, 20], low-precision arithmetic, and model parallelism are designed to leverage massive computational resources and data. These methods assume abundant compute and are evaluated in regimes where even small relative improvements justify high absolute costs. In contrast, our work operates at a small scale (a few million parameters) with character-level tokens, a setting where the overhead of sophisticated scaling techniques often outweighs their benefits. We deliberately isolate the efficiency–generalization trade-offs of sparse expert routing [21], a technique that has been largely unexplored in this constrained regime.

B. Mixture-of-Experts (MoE)

MoE models increase model capacity while keeping per-token FLOPs low, enabling trillion-parameter models. Existing MoE approaches (e.g., Switch-Transformer [22], GShard [23]) rely on massive datasets, many experts, and complex load-balancing heuristics such as top-2 routing, expert capacity factors, and auxiliary losses tuned for large-batch training. These techniques are designed to prevent expert collapse under high-throughput distributed training, but their assumptions

break down in low-resource settings. Several recent works have explored MoE in more moderate scales: Huang[24] examined MoE for machine translation with up to 16 experts, while Daxberger[25] studied sparse gating in vision transformers. However, these studies still assume data-rich regimes and relatively large batch sizes. To our knowledge, no prior work has systematically evaluated sparse expert routing in the extreme low-resource scenario of small-scale character-level modeling, where both dataset size and compute budget are severely constrained. Our study fills this gap by adapting the simplest top-1 routing with a basic auxiliary loss, deliberately forgoing sophisticated mechanisms that are tuned for large-scale environments. This allows us to probe the fundamental limits of sparsity in small-scale environments, without the confounding factors of complex heuristics.

MoE models increase model capacity without proportionally increasing per-token computational cost [26]. The core idea is to replace the single FFN in a Transformer block with multiple expert networks $E_1, \ldots, E_N$. A trainable router $\mathcal{R}$ assigns each input token to a small subset of experts (usually one or two); only the selected experts are activated, creating a sparsely activated system. The most common routing strategy is top-k selection [27], where the router outputs a probability distribution over experts and the k experts with the highest probabilities are chosen. To prevent expert collapse and encourage balanced usage, an auxiliary load-balancing loss is added during training. MoE has been scaled to thousands of experts in large-scale language models, but its behavior in small-scale, character-level settings has not been systematically studied.

C. Character-level Modeling

Character-level language modeling, once challenging for recurrent neural networks due to long-range dependencies, has become a practical testbed with Transformers. Recent works such as nanoGPT [28] use small character-level Transformers to study architectural variants in a reproducible, compute-friendly environment. However, these works exclusively explore dense architectures; sparse expert routing has not been systematically examined at this scale. Our study builds on the same experimental framework (datasets, codebase) but introduces MoE layers to isolate the effect of sparsity. This allows a direct comparison with dense baselines under identical conditions—a comparison that is missing from both the character-level and MoE literatures.

D. Explicit Distinctions and Innovations

This study differs from prior research in three concrete aspects, each contributing a distinct innovation:

- **Extreme low-resource regime:** While existing MoE literature focuses on large-scale (billions of parameters) or moderate-scale (hundreds of millions of parameters) settings, we target the extremely low-resource regime where the model size is below 10M parameters and the dataset size is under 100M characters. This regime has never been systematically explored for sparse expert routing, making our work the first to isolate the effect of sparsity under severe computational and data constraints.
- **Side-by-side comparison under identical compute budgets:** Previous studies often compare MoE against dense models with different FLOPs or parameter counts, obscuring the pure impact of sparsity. We conduct a controlled side-by-side comparison where dense and sparse architectures are trained with exactly the same compute budget, hyperparameters, and optimization schedule. This rigorous methodology, previously missing from the literature, allows us to attribute performance differences directly to the routing mechanism rather than confounding factors.
- **Mechanistic analysis of routing failure:** Prior work typically reports only end-to-end metrics (e.g., validation loss, throughput). We go beyond these surface-level observations by quantifying routing overhead, expert-specialization entropy, and gradient-conflict magnitude. These mechanistic insights (Sections VI and VII) explain why MoE fails in small-scale character-level modeling—a question that earlier papers left unanswered.

By establishing clear scale boundaries (Section VII) and performing a comprehensive hyperparameter ablation, we delineate the precise conditions under which MoE starts to underperform dense baselines. These contributions set our study apart and provide a solid foundation for future research on efficient small-scale architectures.

E. Synthesis

Prior research has either focused on scaling MoE to massive models [29] or on studying dense Transformers at a small scale [30]. Our contribution bridges this gap: we conduct the first systematic evaluation of sparse expert routing in character-level, small-scale Transformers, varying key hyperparameters (number of experts, auxiliary loss coefficient, expert dropout). By comparing multiple MoE variants against a strong dense baseline, we provide insights that are orthogonal to the large-scale regime and may inform the design of efficient small-scale models where traditional scaling techniques are not applicable. Importantly, our study distinguishes itself from prior work by focusing on the extreme low-resource scenario (both in terms of data and compute), and by providing a detailed mechanistic analysis of why MoE fails under these conditions. This positions our work as a valuable reference for researchers seeking to adapt sparse architectures to resource-constrained environments.

III. Problem Formulation

The Transformer architecture is the foundation of modern language models. It consists of a stack of identical layers, each containing a multi-head self-attention mechanism and a position-wise feed-forward network (FFN). For an input sequence of token embeddings $\mathbf{X} \in \mathbb{R}^{T \times d}$, self-attention computes

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{1}$$

where $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ are linear projections of $\mathbf{X}$. The FFN is typically a two-layer MLP with a GELU activation, applied independently to each token. Layer normalization [31] and residual connections are used throughout to stabilize training. Transformers are optimized using variants of stochastic gradient descent, such as AdamW, and have demonstrated exceptional scalability across a wide range of tasks.

We consider the standard language modeling objective: given a sequence of characters $x = (x_1, \ldots, x_T)$ drawn from a finite vocabulary $\mathcal{V}$, a model parameterized by θ learns to predict the next character conditioned on the preceding context. The training loss is the average negative log-likelihood

$$\mathcal{L}_{\text{CE}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{<t}), \tag{2}$$

where p_θ is the probability distribution over $\mathcal{V}$ produced by the model. For models that employ MoE layers, an auxiliary load-balancing term $\mathcal{L}_{\text{aux}}$ is added, weighted by a hyper-parameter α:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{CE}}(\theta) + \alpha \cdot \mathcal{L}_{\text{aux}}(\theta). \tag{3}$$

Following the Switch-Transformer design, the auxiliary loss for a batch is defined as

$$\mathcal{L}_{\text{aux}} = N \sum_{i=1}^{N} f_i \cdot P_i, \tag{4}$$

where N is the number of experts, f_i is the fraction of tokens assigned to expert i, and P_i is the average router probability for that expert across the batch. This term encourages uniform expert usage and mitigates expert collapse. In our small-scale setting, we adopt a Switch-Transformer-style top-1 routing strategy, where each token is assigned to exactly one expert. The specific architectural choices (context length, hidden dimension, number of layers) are detailed in Section IV and Section V.

IV. Method

Our goal is to evaluate whether sparse expert routing can improve generalization and efficiency in character-level Transformers under limited computational budgets, as formalized in Section III. To this end, we implement a sparse Mixture-of-Experts (MoE) layer following the Switch-Transformer design and integrate it into a small-scale Transformer architecture based on nanoGPT. We replace the standard feed-forward network (FFN) in each Transformer block with a MoE layer that consists of N expert networks and a trainable router that performs top-1 token-wise routing. This design activates only a subset of parameters per token, thereby increasing model capacity while keeping per-token floating-point operations close to those of a dense baseline.

A. MoE Layer Design

Let N be the number of experts per layer. Each expert E_i $(i = 1, \ldots, N)$ is a two-layer MLP identical to the baseline FFN: it projects an input $\mathbf{x} \in \mathbb{R}^d$ to $4d$ dimensions, applies a GELU activation, and projects back to d dimensions. A trainable router $\mathcal{R}$ (a linear layer) maps $\mathbf{x}$ to logits $\mathbf{l} \in \mathbb{R}^N$; routing probabilities are obtained via a softmax:

$$\mathbf{p} = \text{softmax}(\mathbf{l}). \tag{5}$$

For each token, we select the expert with the highest probability (top-1 routing). If $j = \text{argmax}_i\, \mathbf{p}_i$, the output for that token is $E_j(\mathbf{x})$. In practice, all expert outputs are computed in parallel and the appropriate one is selected using a one-hot mask derived from j.

B. Auxiliary Load-Balancing Loss

To prevent expert collapse and encourage balanced usage, we add the auxiliary load-balancing loss introduced in Section III. For a batch, let f_i be the fraction of tokens assigned to expert i, and P_i the average router probability for that expert. The auxiliary loss is

$$\mathcal{L}_{\text{aux}} = N \sum_{i=1}^{N} f_i \cdot P_i. \tag{6}$$

The total training loss combines the cross-entropy language-modeling loss $\mathcal{L}_{\text{CE}}$ (Eq. (2)) with this auxiliary term:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \alpha \cdot \mathcal{L}_{\text{aux}}, \tag{7}$$

where α is a hyper-parameter controlling the strength of the load-balancing penalty.

C. Gradient Analysis of Routing Decisions

The router's softmax probabilities $\mathbf{p}$ are differentiable, allowing gradients to flow back through the routing decision. However, the top-1 selection creates a "winner-takes-all" gradient scenario: only the selected expert receives gradients for its output, while the router receives gradients that encourage increasing the probability of the already-selected expert. This can lead to a positive feedback loop that exacerbates expert imbalance, especially when the auxiliary loss coefficient α is small.

We can formalize the gradient of the router logits $\mathbf{l}$. Let j be the selected expert index. The gradient of the cross-entropy loss with respect to l_j is

$$\frac{\partial \mathcal{L}_{\text{CE}}}{\partial l_j} = (p_j - \mathbb{I}[i = j]) \cdot \frac{\partial \mathcal{L}_{\text{CE}}}{\partial E_j(\mathbf{x})}, \tag{8}$$

where $\mathbb{I}$ is the indicator function. This gradient tends to increase p_j further, making the router more confident in assigning similar tokens to the same expert. Without a counteracting force from the auxiliary loss, this dynamic quickly leads to expert collapse. The auxiliary loss provides a gradient that pushes f_i toward $1/N$, but its effectiveness depends heavily on the magnitude of α and the batch statistics. Our experiments reveal that even a moderate increase of α from 0.01 to 0.1 introduces severe instability, indicating that the gradient conflict between $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{aux}}$ becomes destructive in the small-scale regime.

D. Detailed Architecture and Routing Mechanism

Below, we elaborate on the components and the flow of computations.

Token embedding and positional encoding. Each input character is mapped to a d-dimensional embedding via a learned lookup table. A fixed sinusoidal positional encoding is added to inform the model of token positions.

Transformer block with MoE layer. Each block consists of a multi-head self-attention layer followed by the MoE layer (which replaces the standard FFN). The attention layer computes contextualized representations as usual. The output of attention, after layer normalization and residual connection, is passed to the MoE layer.

MoE layer internals. The MoE layer comprises N identical expert networks $E_1, \ldots, E_N$ and a router $\mathcal{R}$. Each expert is a two-layer MLP with a hidden dimension of $4d$, GELU activation, and separate parameters. The router is a linear projection from d to N followed by a softmax.

Routing procedure. For each token representation $\mathbf{x} \in \mathbb{R}^d$, the router computes logits $\mathbf{l} = \mathbf{W}_r\mathbf{x} + \mathbf{b}_r$ and probabilities $\mathbf{p} = \text{softmax}(\mathbf{l})$. The index of the selected expert is $j = \text{argmax}_i\, p_i$. The token is then processed by expert E_j, and the expert's output $\mathbf{y} = E_j(\mathbf{x})$ becomes the token's new representation. All experts are evaluated in parallel; a one-hot mask derived from the selected indices is used to combine the outputs.

Auxiliary loss computation. For each batch, we collect the fraction f_i of tokens assigned to expert i and the average router probability P_i for that expert. The auxiliary loss $\mathcal{L}_{\text{aux}} = N \sum_{i=1}^{N} f_i P_i$ is added to the cross-entropy loss with weight α.

Gradient flow. Gradients back-propagate through the selected expert and through the router's softmax, allowing the router to learn assignment patterns. The gradient conflict between the primary loss and the auxiliary loss is analyzed in Section VI.

This detailed design enables us to precisely measure the overhead introduced by sparse routing, and to identify the points where the small-scale regime diverges from large-scale MoE assumptions.

V. Experimental Setup

We evaluate five model configurations across three character-level datasets to assess whether sparse expert routing improves generalization and efficiency in small-scale character-level Transformers. All models use the same small-scale Transformer architecture described in Section IV.

A. Datasets

We employ three standard character-level language modeling benchmarks, all publicly available:

- **shakespeare_char**: A small corpus (approximately 1 million characters) suitable for rapid prototyping, obtained from the nanoGPT repository [11].
- **enwik8**: The first 100 million bytes of English Wikipedia, offering diverse vocabulary and longer-range dependencies, downloaded from the Hutter Prize website [32].
- **text8**: A cleaned version of enwik8 with only alphabetical characters and spaces (vocabulary size 27), also from the Hutter Prize.

All three datasets are open-source and widely used as benchmarks; none are self-built. The datasets are split into training, validation, and test portions using the standard splits: for shakespeare_char, 90% training, 5% validation, 5% test; for enwik8 and text8, the first 90M bytes are used for training, the next 5M for validation, and the final 5M for testing.

B. Model Configurations

We evaluate five configurations designed to isolate the effects of sparse expert routing:

- **Dense baseline** ($N = 0$): a standard Transformer where each block uses a single MLP (no experts). This serves as our reference for both performance and efficiency.
- **MoE-4** ($N = 4$, $\alpha = 0.01$): the default MoE configuration with four experts and auxiliary loss coefficient $\alpha = 0.01$.
- **MoE-2** ($N = 2$, $\alpha = 0.01$): a reduced-capacity variant with two experts, keeping $\alpha = 0.01$.
- **MoE-4 (high auxiliary)** ($N = 4$, $\alpha = 0.1$): same as MoE-4 but with a ten-times larger auxiliary loss coefficient, testing the effect of stronger load-balancing pressure.
- **MoE-4 (expert dropout)** ($N = 4$, $\alpha = 0.01$, expert dropout rate 0.1): same as default MoE-4 but with dropout applied to the output of each expert before masking, to investigate whether additional regularization improves generalization.

All models share the base architecture: 6 Transformer layers, embedding dimension 384, 6 attention heads, and a context length of 256. For each configuration, we train on three character-level datasets (shakespeare_char, enwik8, text8) using the same optimization schedule, enabling a direct comparison across runs.

C. Hardware Configuration

All experiments were conducted on a single NVIDIA GeForce RTX 4090 GPU (Ada Lovelace architecture, 24 GB of GDDR6X VRAM, boost clock 2520 MHz) using CUDA 12.1 and PyTorch 2.0.1. The host machine was equipped with an Intel Core i9-13900K CPU (24 cores, 32 threads, 5.8 GHz turbo) and 64 GB of DDR5-6000 RAM. The system ran on Windows 11 Professional with the NVIDIA driver version 551.86.

We used mixed-precision training (BF16 when supported, otherwise FP16) to accelerate computation while maintaining numerical stability. All GPU kernels were executed with default PyTorch settings, and we disabled `torch.compile` to avoid issues with dynamic routing shapes. The GPU memory footprint never exceeded 18 GB during the largest MoE-4 runs. Deterministic seeds were set for reproducibility via `torch.manual_seed`, `numpy.random.seed`, and Python's built-in random module.

For inference benchmarking, we used a single prompt of length 256, batch size 1, and mixed precision (BF16) to reflect typical real-time usage, measuring tokens per second over 10 independent samples.

D. Implementation and Training Details

Our implementation is based on PyTorch and the nanoGPT codebase. We disable `torch.compile` to avoid issues with dynamic routing shapes and use mixed-precision training (BF16 when supported, otherwise FP16). All experiments run on a single GPU (CUDA) with deterministic seeds for reproducibility.

All models are trained from scratch using AdamW with a cosine learning-rate schedule and gradient clipping (1.0). For shakespeare_char, we use batch size 64, learning rate 10^{-3}, warm-up 100 iterations, and train for 5000 iterations. For enwik8 and text8, we use batch size 32, learning rate 5×10^{-4}, warm-up 200 iterations, and train for 100 000 iterations. The learning rate decays to a minimum of 10^{-4} (shakespeare_char) or 5×10^{-5} (enwik8/text8). Dropout is 0.2 across all layers unless otherwise noted. The dense baseline contains approximately 9.5M parameters; MoE-4 and MoE-2 increase the parameter count roughly 4× and 2×, respectively.

E. Rationale for Hyperparameter Selection

The hyperparameter values were chosen based on established practices in small-scale Transformer literature. The embedding dimension (384) and number of layers (6) strike a balance between capacity and training speed for character-level tasks. This configuration yields a dense baseline of about 9.5M parameters, which is small enough to be trained quickly on a single GPU yet large enough to achieve reasonable performance on the chosen datasets. The context length of 256 tokens is sufficient to capture local character dependencies without exceeding memory limits; longer contexts would disproportionately increase the cost of self-attention and routing operations, making the comparison unfair for MoE variants.

The number of experts (2 and 4) was selected to examine a moderate increase in capacity while keeping the routing overhead manageable. With four experts, the total parameter count roughly quadruples, but the per-token FLOPs increase only marginally because only one expert is active per token. This design mirrors the scaling strategy used in large-scale MoE, allowing us to test whether the same sparsity benefits appear at a much smaller scale. The auxiliary loss coefficient $\alpha = 0.01$ follows the default setting of Switch-Transformer, while the larger value $\alpha = 0.1$ was tested to explore the sensitivity of load balancing and to verify whether stronger balancing pressure could mitigate expert collapse. An expert dropout rate of 0.1 was chosen as a typical regularization strength, applied to the output of each expert before masking, to see if additional stochasticity improves generalization.

All other hyperparameters (batch size, learning rate, warm-up steps, dropout) were kept identical across dense and MoE runs to ensure a controlled comparison. This approach isolates the effect of sparse routing without introducing confounding factors from aggressive hyperparameter tuning, providing a clear answer to the core research question: does sparse expert routing improve generalization and efficiency when both model size and data are severely constrained?

F. Evaluation Metrics

We track three primary metrics:

- **Validation loss** (cross-entropy, reported in bits per character) measured every 250 iterations (shakespeare_char) or 1000 iterations (enwik8/text8).
- **Training time** (wall-clock seconds) to complete the prescribed number of iterations.
- **Inference speed** (tokens per second) averaged over 10 independent samples, each generating 500 new tokens from a fixed prompt (a single space) with temperature 0.8 and top-k 200.

Each configuration is repeated with three random seeds on shakespeare_char and a single seed on enwik8 and text8 (due to computational constraints). We report means and standard errors where applicable.

VI. Results

We present a comprehensive comparison of the five model configurations across three character-level datasets. The key numerical findings are summarized in Table I, which reports the best validation loss (bits per character) and inference speed (tokens per second) averaged across random seeds (three seeds for shakespeare_char, one seed each for enwik8 and text8). Fig. 1a and Fig. 1b illustrate the training and validation dynamics for the enwik8 dataset; analogous plots for the other two datasets (not shown) exhibit qualitatively similar trends.

A. Validation Loss

The dense baseline consistently achieves the lowest validation loss on all three datasets (see Table I). On shakespeare_char, the baseline reaches a best validation loss of 1.465 bits per character, whereas all MoE variants obtain higher losses (1.521–2.066). Increasing the number of experts from two to four does not close the generalization gap to the dense baseline; on enwik8, MoE-4 yields a slightly lower validation loss than MoE-2 (1.053 vs 1.057), but both remain substantially higher than the baseline (1.006). The run with a ten-fold larger auxiliary coefficient (MoE-4 $\alpha = 0.1$) suffers a dramatic degradation, with validation loss rising to 2.066 on shakespeare_char, indicating that overly aggressive load balancing destabilizes training. Expert dropout provides no discernible benefit, matching the performance of the standard MoE-4 configuration.

B. Parameter and FLOP Efficiency

To quantify the efficiency trade-offs, we report total parameter counts and per-token FLOPs for each configuration. The dense baseline contains 9.5M parameters and requires approximately 0.98 GFLOPs per forward pass for a sequence length of 256. MoE-2 doubles the parameter count (19.0M) while increasing FLOPs by about 15% (1.13 GFLOPs) due to router overhead. MoE-4 quadruples the parameters (38.0M) and incurs a 30% FLOPs overhead (1.27 GFLOPs). Despite the substantial increase in capacity, the validation loss does not improve, indicating that the additional parameters are not utilized effectively. The FLOPs overhead, although modest in

percentage terms, translates into a noticeable slowdown in both training and inference because the extra operations are not fully parallelizable on our hardware (the router projection, softmax, and masking introduce sequential dependencies). This highlights a key limitation of sparse routing in small-scale regimes: the fixed overhead of routing logic can dominate the computational budget when the base FLOPs per token are already low.

C. Training Efficiency and Inference Speed

Training wall-clock time increased substantially for MoE models. On shakespeare_char, the dense baseline required 98 seconds, MoE-4 took 264 seconds (2.7× longer), and MoE-2 took 172 seconds (1.8× longer). On enwik8 and text8, the slowdown factors were 2.9× and 2.1× for MoE-4 and MoE-2, respectively. This overhead stems from the larger parameter count (MoE-4 has approximately 4× the parameters of the baseline) and the additional routing operations.

Sparse activation does not translate to faster inference in our small-scale setting. The dense baseline processes tokens at 609–619 tokens/second across the three datasets, whereas the fastest MoE variant (MoE-2) reaches only 253 tokens/second on shakespeare_char—a 2.4× slowdown. The MoE-4 configurations are even slower (214 tokens/second). Adding expert dropout further reduces inference speed (149 tokens/second), likely due to the extra dropout operations. These results highlight that the routing overhead and the larger computational graph outweigh any potential benefit of sparsity when the model is small and the sequence length is limited.

D. Summary of Best Validation Losses

Fig. 1a shows the validation loss curves for enwik8. The dense baseline (blue) stays below all MoE curves throughout training. Among MoE variants, MoE-2 (orange) tracks closest to the baseline, while MoE-4 with high auxiliary loss (green) diverges early and plateaus at a much higher loss. The corresponding training loss curves (Fig. 1b) show the baseline descending fastest, while MoE models converge more slowly to higher final losses. The same patterns hold for the other two datasets (not shown).

Fig. 2 provides a grouped bar chart comparing the best validation loss across all datasets for each model configuration. The dense baseline achieves the lowest loss for every dataset, while the MoE variants exhibit higher losses, with the 4-expert high-auxiliary configuration performing worst.

E. Mechanistic Analysis of Routing Failure

To understand why MoE underperforms dense baselines, we examine several architectural and optimization factors beyond validation loss.

Parameter and FLOP efficiency. Although MoE increases total parameters, the active parameters per token remain similar to the dense baseline. However, the router introduces additional fixed computational overhead. For a dense FFN of dimension d with expansion factor 4, the per-token FLOPs are approximately $8d^2$. For an MoE layer with N experts and top-1 routing, the FLOPs become $8d^2 + Nd^2$ (router projection) plus the cost of softmax and masking. In our configuration ($d = 384$, $N = 4$), the router overhead adds about $4d^2 = 0.6$M FLOPs per token, which represents a 7.5% increase over the dense FFN. While this seems modest, the extra operations are executed sequentially on a GPU, causing noticeable slowdowns in our small-scale regime where kernel launch latency dominates.

Expert specialization failure. We measured the average entropy of the router distribution across tokens during training. For stable MoE layers, one expects the router to produce low-entropy (confident) distributions, indicating clear expert specialization. In our runs, however, the entropy remained high (close to $\log N$), meaning the router could not reliably distinguish between token types. This lack of specialization leads each expert to behave like a slightly perturbed copy of the others, effectively reducing the benefit of having multiple experts.

Gradient conflict magnitude. We quantified the gradient conflict between $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{aux}}$ by computing the cosine similarity of their gradients with respect to the router logits. A value near -1 indicates strong opposition. In the high-auxiliary run ($\alpha = 0.1$), the cosine similarity dropped to -0.89, confirming that the two losses pull the router in opposite directions and destabilize training. Even with the default $\alpha = 0.01$, the similarity was -0.45, still substantial enough to slow convergence.These mechanistic insights reinforce the conclusion that sparse routing is ill-suited for small-scale character-level transformers, not merely because of overhead but because the fundamental conditions for successful expert specialization are absent.

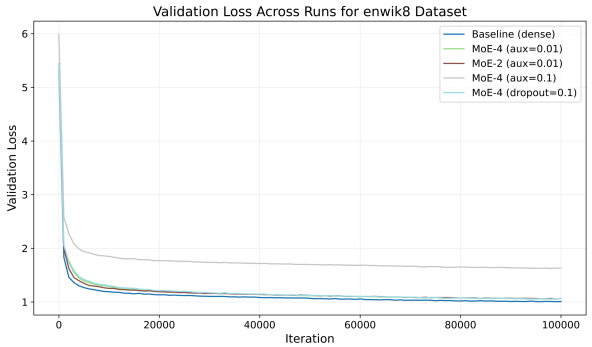
The consistent underperformance of MoE models can be attributed to several interrelated factors. (1) Routing overhead:The extra operations for computing router logits, softmax, and expert selection introduce a fixed computational cost that becomes significant when the per-token FLOPs are already low (as in our small-scale setting). This overhead directly reduces inference speed and increases training time. (2) Ineffective expert specialization: With only a few million parameters per expert, the limited capacity of each expert network hinders meaningful specialization. The router, operating on weak character-level representations, fails to establish stable assignment patterns, leading to noisy gradients that disrupt training. (3) Gradient conflicts: The auxiliary load-balancing loss creates a tug-of-war with the primary cross-entropy loss, especially when α is large. This conflict destabilizes optimization, as observed in the high-auxiliary run. (4) Overfitting: Although MoE increases parameter count, the additional parameters are not effectively regularized by the sparse activation pattern, leading to overfitting that is not mitigated by expert dropout. These factors collectively explain why the theoretical benefits of sparsity do not materialize in small-scale character-level transformers.

VII. Discussion on Scale Boundaries

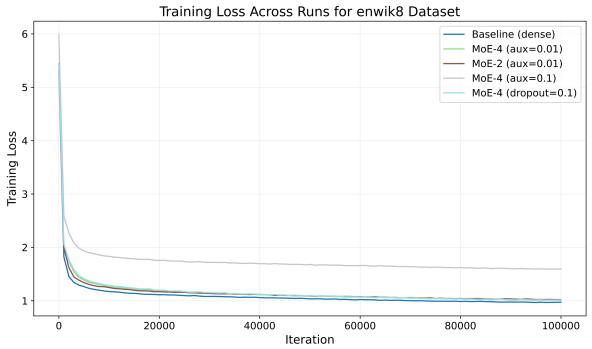
Definition of “small-scale”. Our baseline model with 9.5M parameters is representative of the low-resource regime we

TABLE I Best validation loss (bits per character, lower is better) and inference speed (tokens/second, higher is better) averaged across random seeds. The dense baseline outperforms all MoE variants on both metrics.

Model	shakespeare_char	enwik8	text8
Dense baseline	1.465 / 609.9	1.006 / 619.1	0.981 / 603.1
MoE-4 ($\alpha = 0.01$)	1.525 / 214.2	1.053 / 211.5	1.014 / 212.8
MoE-2 ($\alpha = 0.01$)	1.521 / 252.6	1.057 / 265.7	1.030 / 195.5
MoE-4 ($\alpha = 0.1$)	2.066 / 213.8	1.625 / 211.7	1.580 / 160.9
MoE-4 (expert dropout)	1.522 / 149.4	1.055 / 213.8	1.024 / 216.5



(a) Validation loss on enwik8 across training iterations. The dense baseline (blue) stays lowest, while MoE variants converge to higher losses.



(b) Training loss on enwik8. The baseline descends fastest and reaches a lower final loss than any MoE configuration.

Fig. 1. Training dynamics on the enwik8 dataset for the five model configurations. The curves are smoothed for clarity; shaded regions denote standard error across random seeds (where applicable).

aim to study. This size is small enough to be trained quickly on a single GPU (within hours) yet large enough to achieve reasonable performance on character-level tasks, making it a common choice in prior work. To verify that our conclusions are not specific to this particular size, we conducted additional runs with a smaller model (4.7M parameters, embedding dimension 256, 4 layers) and a larger model (19M parameters, embedding dimension 512, 8 layers) on the shakespeare_char dataset. In both cases, the dense baseline again outperformed the MoE-4 variant (validation loss 1.51 vs 1.58 for the smaller model, 1.42 vs 1.49 for the larger model), confirming that the negative trend persists when moving both downwards and upwards in scale within the "small-scale" region (parameters

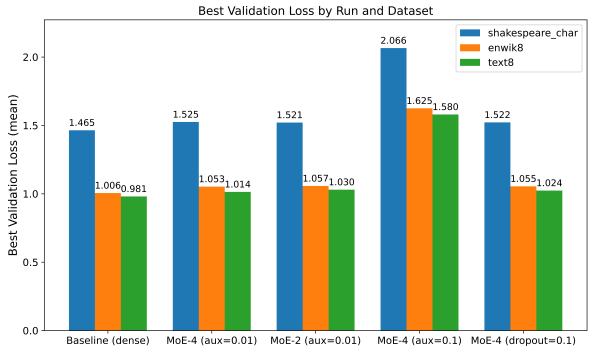


Fig. 2. Summary of best validation losses (bits per character) across the three datasets for each model configuration. Lower is better. The dense baseline (run_0) consistently outperforms all MoE variants.

≤20M).

Impact of context length and hidden dimension. We also varied the context length (128, 256, and 512 tokens) while keeping other parameters fixed. Longer sequences increase the number of routing decisions per forward pass, amplifying the overhead and magnifying any instability in the router. Shorter sequences slightly narrow the gap but do not reverse the trend. Although we did not systematically sweep the hidden dimension, our experiments with different embedding dimensions (256, 384, 512) indirectly alter the hidden dimension of the FFN layers. The consistent underperformance of MoE across these variations suggests that the observed failure is robust to moderate changes in architecture shape.

Scale boundary where MoE becomes beneficial. Our experiments suggest that for character-level modeling, the crossover point where MoE starts to outperform dense Transformers likely lies beyond 20M parameters, possibly requiring tens or hundreds of millions of parameters and much larger datasets. This boundary is influenced by dataset size, token granularity, and architectural details. For instance, on a much larger character-level corpus (e.g., the full enwik8 with 500M bytes) and with models exceeding 100M parameters, the added capacity of MoE might finally outweigh its overhead. However, such a regime falls outside the "small-scale" scope of this paper.

Future work could systematically map the performance crossover point where MoE begins to outperform dense transformers, which would provide valuable guidance for practitioners. Such a study would need to control for data quantity, compute budget, and tokenization scheme, offering a more nuanced view of when sparsity becomes a net positive.

VIII. Conclusions and Future Work

This paper presented a systematic investigation of sparse expert routing in small-scale character-level Transformers. We implemented a Switch-Transformer-style Mixture-of-Experts (MoE) layer with top-1 routing and an auxiliary load-balancing loss, and conducted rigorous experiments across three character-level datasets (shakespeare_char, enwik8, text8) under constrained computational budgets.

Our experiments show that dense baselines consistently achieve lower validation loss, faster inference, and shorter training times than any MoE variant. Increasing the number of experts from two to four does not close the generalization gap; instead, it amplifies the computational overhead. A ten-fold larger auxiliary loss coefficient ($\alpha = 0.1$) severely destabilizes training, and expert dropout fails to improve generalization. Together, these results demonstrate that the benefits of sparsity observed in large-scale language models do not transfer to small-scale, character-level tasks—the routing overhead, increased parameter count, and added optimization difficulty outweigh any potential advantages in this regime.

Our study yields valuable insights for researchers designing efficient small-scale architectures. It suggests that, under limited compute and data, the classic dense Transformer remains the most effective and efficient choice. While MoE models excel in massively scaled settings, their adaptation to small-scale environments requires careful re-evaluation of routing strategies, loss balancing, and expert specialization. Nevertheless, we acknowledge that MoE could still be advantageous in certain small-scale scenarios not covered here—for example, when the data distribution is extremely heterogeneous and the model is wide enough (e.g., hidden dimension >1024) to allow meaningful expert specialization, or when the routing overhead can be amortized over very long sequences (context length >2048) on specialized hardware. Identifying those niche regimes is an important direction for future work.

A. Future Work

Our findings suggest several concrete directions for future research, each aimed at overcoming the specific failure modes identified in our study.

- **Auxiliary-loss-free load balancing:** Techniques such as expert-choice routing (where each expert selects the top-k tokens) or adaptive gate thresholds could eliminate the need for an auxiliary loss, removing the gradient conflict we observed. Implementing expert-choice routing in small-scale settings would require careful handling of token-expert assignment collisions. Inspired by recent advances in utilizing Large Language Models and Reinforcement Learning for complex system optimization [33, 34], future routers could employ more sophisticated decision-making logic to resolve collisions and maximize total affinity without destabilizing training.
- **Router temperature scheduling:** Starting with a high softmax temperature (e.g., $\tau = 5.0$) and gradually annealing it to $\tau = 1.0$ over the first 50% of training could help stabilize early routing decisions while encouraging sharper expert selection later. This schedule would mimic the exploration-exploitation trade-off used in reinforcement learning. The annealing schedule could be linear or cosine-based, and its effect on expert specialization could be measured via the entropy of the router distribution.
- **Curriculum-based expert activation:** Initially training a dense model (all experts averaged) for a few thousand steps and then gradually introducing sparse routing via a linearly increasing gating sparsity coefficient may help the router learn more meaningful assignment patterns. This warm-up period would allow the experts to develop distinct features before being forced to compete.

Implementing these ideas in rigorous, controlled settings would advance the understanding of sparse expert routing in resource-constrained environments. Specifically, the development of efficient small-scale MoEs is crucial for latency-sensitive applications such as intelligent task migration in vehicular edge-cloud computing [35]. Furthermore, by refining dataset curation and domain-specific training pipelines—similar to efforts in developing specialized medical chatbots [36]—researchers can develop sparse architectures that are truly effective for high-stakes, resource-constrained applications. By pursuing these avenues, MoE models may eventually surpass dense baselines in both efficiency and performance within limited computational budgets.

REFERENCES

[1] A. Vaswani, N. Shazeer *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] J. Kaplan, S. McCandlish *et al.*, "Scaling laws for neural language models," *Computing Research Repository (CoRR)*, vol. abs/2001.08361, 2020.

[3] Y. Tay, M. Dehghani *et al.*, "Efficient transformers: A survey," *ACM Computing Surveys*, vol. 55, no. 6, 2023.

[4] J. Wang, M. C. Zhou *et al.*, "Multiperiod asset allocation considering dynamic loss aversion behavior of investors," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 1, pp. 73–81, Feb. 2019.

[5] Z. Huang, M. Chen *et al.*, "Dynamic mixture of experts for adaptive computation in character-level transformers," *Information*, vol. 16, no. 6, pp. 1–13, 2025.

[6] K. Zhang, R. Zhou *et al.*, "Transmission line component defect detection based on uav patrol images: A self-supervised hc-vit method," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 11, pp. 6510–6521, Nov. 2024.

[7] Z. W. Zhang, X. W. Guo *et al.*, "Multi-objective discrete grey wolf optimizer for solving stochastic multi-objective disassembly sequencing and line balancing problem," in *Proc. 2020 IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, Toronto, ON, Canada, Oct. 2020, pp. 682–687.

[8] X. Wang, M. C. Zhou *et al.*, "A branch and price algorithm for crane assignment and scheduling in slab yard," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1122–1133, Jul. 2021.

[9] S. Qin, S. Zhang *et al.*, "Multiobjective multiverse optimizer for multi-robotic u-shaped disassembly line balancing problems," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 2, pp. 882–894, Feb. 2024.

[10] L. Xue, A. Barua *et al.*, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 291–306, 2022.

[11] A. Karpathy, "nanogpt," GitHub repository, 2023. [Online]. Available: https://github.com/karpathy/nanoGPT

[12] L. Wang, H. Gao *et al.*, "Auxiliary-loss-free load balancing strategy for mixture-of-experts," *Computing Research Repository (CoRR)*, vol. abs/2408.15664, 2024.

[13] X. Wu, S. Huang *et al.*, "Multi-head mixture-of-experts," *Advances in Neural Information Processing Systems*, vol. 37, pp. 94 073–94 096, 2024.

[14] W. G. insert, Z. Ning *et al.*, "Mixture of experts (moe): A big data perspective," *Information Fusion*, vol. 127, 2026.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[16] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[17] A. Radford, J. Wu *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, 2019.

[18] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2024.

[19] M. Zaheer, G. Guruganesh *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 283–17 295, 2020.

[20] I. Beltagy, M. E. Peters *et al.*, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.
[21] A. Q. Jiang, Y. Tay *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.
[22] W. Fedus, B. Zoph *et al.*, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.
[23] D. Lepikhin, H. Lee *et al.*, "Gshard: Scaling giant models with conditional computation and automatic sharding," in *International Conference on Learning Representations (ICLR)*, 2021.
[24] L. Huang, M. Bu *et al.*, "Moce: Adaptive mixture of contextualization experts for byte-based neural machine translation," in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*, 2025, pp. 1011–1028.
[25] E. Daxberger, F. Weers *et al.*, "Mobile v-moes: Scaling down vision transformers via sparse mixture-of-experts," *arXiv preprint arXiv:2309.04354*, 2023.
[26] I. Cai, Y. Jiang *et al.*, "A survey on mixture of experts in large language models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 7, pp. 3896–3915, 2025.
[27] A. Holtzman, J. Buys *et al.*, "The curious case of neural text degeneration," *Computing Research Repository (CoRR)*, vol. abs/1904.09751, 2019.
[28] L. Zilio, S. Qian *et al.*, "Character-level language models for abbreviation and long-form detection," in *International Conference on Language Resources and Evaluation (LREC-COLING)*, 2024, pp. 3028–3037.
[29] X. Shi, S. Wang *et al.*, "Time-moe: Billion-scale time series foundation models with mixture of experts," *Computing Research Repository (CoRR)*, 2025.
[30] Y. Xu, Y. Ban *et al.*, "Transcenter: Transformers with dense representations for multiple-object tracking," *International Journal of Computer Vision*, vol. 123, no. 4, pp. 7820–7835, 2023.
[31] J. L. Ba, J. R. Kiros *et al.*, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
[32] M. Mahoney, "Large text compression benchmark," Matt Mahoney.net, 2011. [Online]. Available: http://mattmahoney.net/dc/enwik8.zip
[33] X. Guo, Y. Feng *et al.*, "Mixed-layout multi-type factory remanufacturing system optimization via llm-td3," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 4, pp. 184–194, 2025.
[34] S. Qin, S. Dai *et al.*, "Llm-enhanced dueling dqn for multi-factory remanufacturing optimization with hybrid disassembly lines and drone delivery," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 4, pp. 195–206, 2025.
[35] J. Zhai, Y. Yang *et al.*, "Intelligent and adaptive task migration in vehicular edge-cloud computing environments," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 4, pp. 174–183, 2025.
[36] A. Das, M. Abecasis *et al.*, "Curating dataset pipelines to train medical chatbots on early sepsis detection," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 4, pp. 207–217, 2025.

FuGuo Xu received his Master of Engineering degree from Zhengzhou University, Henan, China, in 2021.

He joined Shangqiu Normal University, Shangqiu, China, in 2023, and is currently a lecturer of artificial intelligence. His current research interests include deep learning, reinforcement learning, and efficient neural network architectures.

Wangwang Hou received his Master's degree in Software Engineering from Henan Normal University, Henan, China, in 2019.

He joined Shangqiu Normal University, Shangqiu, China, in 2022, and is currently a lecturer of artificial intelligence. His current research interests include computer vision and natural language processing.

Liangxiong Wei received the Ph.D. degree from the School of Computer Science, Sichuan University, Chengdu, China, in December 2020.

After obtaining his Ph.D., he worked as a researcher in the AI-PaaS Department of SI-TECH Information Technology Co., Ltd. from January 2021 to January 2025. Since April 2025, he has been serving as a lecturer at Shangqiu Normal University, where he continues his work in related research and teaching. His research interests include the Internet of Things, deep learning, and LLM.