DTRNet: Dynamic Token Routing Network to Reduce Quadratic Costs in Transformers

Aman Sharma[4*], Saeed Najafi[2*], Parsa Farinneya[5*], Benyamin Jamialahmadi[5*], Marzieh S. Tahaei[6*], Yuhe Fan[1*], Mehdi Rezagholizadeh[7*], Boxing Chen[1], Aref Jafari[3,7*]

[1]Huawei Noah's Ark Lab [2]University of Alberta [3]University of Waterloo [4] Qualcomm [5] Chubb [6] Autodesk [7]Advanced Micro Devices, Inc. (AMD)

{aref.jafari, mehdi.rezagholizadeh}@amd.com

Abstract

Transformers achieve state-of-the-art results across many tasks, but their uniform application of quadratic self-attention to every token at every layer makes them computationally expensive. We introduce DTRNet (Dynamic Token Routing Network), an improved Transformer architecture that allows tokens to dynamically skip the quadratic cost of cross-token mixing while still receiving lightweight linear updates. By preserving the MLP module and reducing the attention cost for most tokens to linear, DTRNet ensures that every token is explicitly updated while significantly lowering overall computation. This design offers an efficient and effective alternative to standard dense attention. Once trained, DTRNet blocks routes only $\sim$10% of tokens through attention at each layer while maintaining performance comparable to a full Transformer. It outperforms routing-based layer skipping methods such as MoD and D-LLM in both accuracy and memory at matched FLOPs, while routing fewer tokens to full attention. Its efficiency gains, scales with sequence length, offering significant reduction in FLOPs for long-context inputs. By decoupling token updates from attention mixing, DTRNet[1] substantially reduces the quadratic share of computation, providing a simple, efficient, and scalable alternative to Transformers.

1 Introduction

Transformers have emerged as the dominant architecture across natural language processing (NLP), vision, and multi-modal tasks, powering state-of-the-art (SOTA) performance in large language models (LLMs) such as GPT-4, Claude, and Gemini (OpenAI, 2023; Anthropic, 2023; Gemini Team, 2024). Despite their empirical success, the quadratic complexity of self-attention poses a significant bottleneck, particularly as sequence lengths and model sizes scale beyond billions of parameters (Vaswani et al., 2017). This limitation has motivated a growing body of work focused on improving the efficiency of Transformers through adaptive computation, token routing, and sparse attention mechanisms (Elhoushi et al., 2024; Wu and Tu, 2024; Raposo et al., 2024; Dehghani et al., 2019; Child et al., 2019).

A key observation in Transformer-based models is that not all tokens require equal compute at every layer (Shin et al., 2025). This is supported further by our layerwise cosine similarity analysis of token embeddings in a dense Transformer (See Section Methodology), which reveals that inner-layer embeddings change only marginally across adjacent layers, highlighting significant redundancy in token updates across layers. These findings suggest that uniformly applying attention to every token at every layer may be inefficient, and that routing-based mechanisms can yield substantial savings by skipping unnecessary computations.

Recent work has explored reducing Transformer computation through adaptive token routing. Mixture-of-Depth (MoD) (Raposo et al., 2024) applies expert-choice routing on alternating layers, selectively choosing which tokens are processed while skipping the rest. In contrast, D-LLM (Jiang et al., 2024) performs token-choice gating at every layer, dynamically deciding whether each token should pass through a Transformer block or bypass it entirely. Despite their differences, both approaches share a key limitation: tokens that bypass a block skip both the self-attention and MLP components, receiving no update. This results in lower representational quality and weaker performance compared to standard Transformers.

Inspired by recent work, highlighting the relative importance of MLP/FFN sublayers and the redundancy of many attention components (Dong et al., 2025; Cao et al., 2024; Yu and Ananiadou, 2023),

[*]Work done while at Huawei.

[1]The code is released at https://github.com/Aman26Sharma/DTRNet

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 47476–47489
October 24-29, 2026 ©2026 Association for Computational Linguistics

we introduce the **Dynamic Token Routing Network (DTRNet)**, a Transformer architecture that retains MLP and reduces the quadratic computation via a two-path design. At each layer, a learned router assigns each token to either (i) a standard *quadratic* full-attention path or (ii) a lightweight *linear* path that updates the token using shared value and output projections. Unlike prior routing-based methods (e.g., MoD, D-LLM) that skip both attention and MLP entirely, our DTR layers retains the MLP for all tokens.

Once trained, DTRNet routes only $\sim$10% of tokens on average through attention at each layer, while achieving performance comparable to a full Transformer. DTRNet outperforms both MoD and D-LLM in terms of accuracy and memory under same computation budget. DTRNet achieves this while routing fewer tokens to full attention compared to MoD and D-LLM, thereby further reducing the quadratic complexity of the Transformer. The advantage of DTRNet becomes even more pronounced as sequence lengths increase, providing significantly lower FLOPS and lower perplexity than the baselines. These findings highlight the effectiveness of DTRNet's projection-based update path, which not only preserves representational quality but also enables a substantial reduction in quadratic computation. This makes DTRNet particularly well-suited for language modeling and long-context applications, where sequences can grow far beyond the training horizon.

Our main contributions are:

- A dynamic routing architecture that enable explicit token updates at reduced cost via a lightweight linear path and a quadratic attention path while retaining a shared MLP block.
- A simple training objective that penalizes routing tokens to attention, allowing the model to learn when attention is needed while preserving accuracy.
- Empirical analysis showing that DTRNet outperforms SOTA performance at comparable compute and improves efficiency across model scales (360M, 1.3B). Our evaluations show that efficiency and FLOPs gain of DTRNet increases with sequence length, surpassing dense Transformers and prior routing methods.

2 Related Work

Sparse Attention Architectures. The quadratic complexity of self-attention in Transformers has spurred research into more efficient architectures. Sparse attention mechanisms, such as those in Longformer (Beltagy et al., 2020), Reformer (Kitaev et al., 2020) and BASED (Arora et al., 2024), reduce computation by limiting attention to local windows or approximating it with kernel-based methods. Unlike these approaches, which modify attention patterns, DTRNet focuses on token-level computation, selectively routing tokens to bypass quadratic attention.

Depth-Adaptive Inference. Depth-adaptive methods aim to reduce inference costs by dynamically adjusting the number of layers processed per input. Methods like LayerSkip (Elhoushi et al., 2024), SortedNet (Valipour et al., 2023), SortedLLaMA (Kavehzadeh et al., 2024) and Balcony (Jamialahmadi et al., 2025) employ different variants of early-exit strategies, enabling intermediate layers to produce predictive outputs. Flextron (Cai et al., 2024) uses a router to select sub-networks of varying depths. These methods, however, apply uniform computation across all tokens in a sequence, lacking fine-grained token-level control. Skipping more than 30% of layers often degrades performance due to this insufficient token-specific adaptivity (Jamialahmadi et al., 2025). Unlike depth-adaptive methods that skip entire layers uniformly for all tokens, DTRNet overcomes this by routing tokens individually, allowing fine-grained computation allocation tailored to each token's needs.

Token-Level Routing. Token-level depth-wise routing offers a more granular approach by dynamically allocating computation per token. Mixture-of-Depths (MoD) (Zhou et al., 2022) introduces a learned router at each layer that selects a top-k subset of tokens for full attention and MLP computation, while the rest are forwarded through residual connections without updates. To support inference-time deployment, an auxiliary network is trained to predict routing decisions. However, this introduces a consistency mismatch between training and inference, as the router behavior changes post-training. This often results in suboptimal token selection and degraded performance, especially in large-scale models.

D-LLM (Jiang et al., 2024) proposes dynamic

token-wise layer skipping for pretrained Transformers by attaching lightweight token routers at each layer. These use Gumbel-Softmax sampling to make discrete routing decisions, with auxiliary losses encouraging partial usage of each layer. While effective at reducing computation, D-LLM suffers from layer starvation, where certain layers, particularly deeper ones, receive limited gradient signal, leading to underutilized capacity and weaker generalization. Because token-level routing methods like MoD and D-LLM operate at the exact same granularity as DTRNet, we consider them to be the most appropriate baselines for isolating and evaluating the specific effects of our dynamic routing mechanism, in contrast to early-exit methods or sparse attention architectures that modify the attention pattern itself.

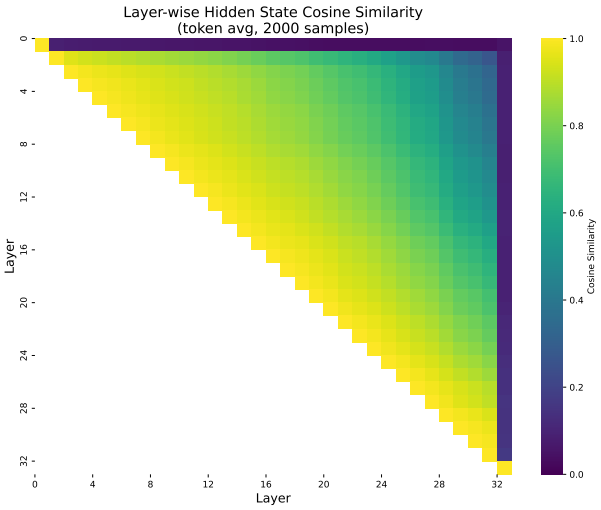


Figure 1: Average Layerwise cosine similarity of token embeddings in a 1.3B SmolLM model

3 Methodology

DTRNet is a Transformer architecture designed to reduce the quadratic complexity of self-attention by dynamically routing tokens through either a full attention path or a lightweight projection-based path. Motivated by empirical evidence of computational redundancy in Transformer token updates, DTRNet ensures all tokens receive meaningful updates while minimizing attention overhead. This section outlines the motivation from token redundancy analysis, the DTRNet layer structure, training strategy, and architectural design choices.

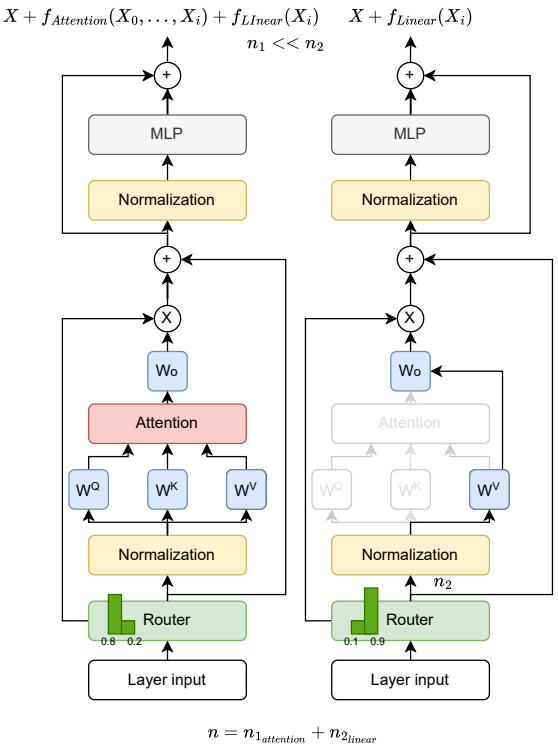


Figure 2: **DTRNet Layer.** Left: tokens routed to the self-attention path undergo full cross-token mixing. Right: tokens routed to the projection-only (bypass) path skip mixing and receive a token-local update via W^V and W^O, followed by the shared FFN. Both paths share parameters

3.1 Motivation: Token Redundancy

To inform DTRNet's design, we analyzed token representation redundancy in a 1.3B-parameter Transformer (SmolLM https://huggingface.co/blog/smollm) evaluated on WikiText (Merity et al., 2016). We computed the average cosine similarity between token embeddings $x^{(l)} \in \mathbb{R}^{n \times d}$ across layers l, where n is the sequence length and d is the hidden dimension. The resulting similarity matrix $S \in \mathbb{R}^{L \times L}$, visualized in Figure 1, reveals high similarity between adjacent inner layers (e.g., $S_{i,i+1} \approx 0.98$), indicating minor representation changes, while the first and last layers show lower similarity with their neighbors. These findings suggest redundancy in adjacent layers showing potential for reduced computation. In contrast, the first and last layers exhibit low similarity with their neighbors, indicating more substantial transformations at the boundaries.

3.2 DTRNet Layer Structure

Each DTRNet layer operates on the input token embeddings $X^{(l)} = [x_1^{(l)}, \ldots, x_n^{(l)}] \in \mathbb{R}^{n \times d}$, where n

is the number of tokens and d is the hidden dimension, and l is the layer index. A learned *router* $\mathcal{G}^{(l)}$ processes each token embedding $x_i^{(l)}$ and outputs a probability distribution over two computational paths:

- the **quadratic path**, which applies standard multi-head self-attention, and
- the **linear path**, which applies linear projections without cross-token interaction.

Token Router. The router $\mathcal{G}^{(l)}$ is a two-layer feedforward network:

$$\mathcal{G}_i^{(l)} = \text{softmax}\left(\text{SiLU}(x_i^{(l)} W_1^{(l)}) \cdot W_2^{(l)}\right) \in \mathbb{R}^2 \tag{1}$$

where $W_1^{(l)} \in \mathbb{R}^{d \times \frac{d}{2}}$, $W_2^{(l)} \in \mathbb{R}^{\frac{d}{2} \times 2}$, and the output $\mathcal{G}_i^{(l)} = [g_{i,\text{attn}}^{(l)}, g_{i,\text{bypass}}^{(l)}]$ represents the soft routing scores.

We perform *hard routing* by selecting the path with the higher score:

$$\delta_i^{(l)} = \begin{cases} 1 & \text{if } g_{i,\text{attn}}^{(l)} > g_{i,\text{bypass}}^{(l)} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Computation Paths. Each token is routed to one of two computational branches:

Quadratic Path: If $\delta_i^{(l)} = 1$, the token is updated via standard multi-head self-attention:

$$y_i^{(l)} = \text{MLP}\left(g_{i,\text{attn}}^{(l)} \cdot \text{Attn}(x_i^{(l)})\right) \tag{3}$$

where:

$$\text{Attn}(x_i^{(l)}) = \left(\text{softmax}\left(\frac{q_i K^\top}{\sqrt{d}} + M_\text{C}\right) V\right) W^O$$
$$q_i = x_i^{(l)} W^Q, \quad K = X_\text{attn}^{(l)} W^K, \quad V = X_\text{attn}^{(l)} W^V \tag{4}$$

and $M_\text{C} \in \mathbb{R}^{n \times n}$ is the causal mask. Notably, to maintain strict compute boundaries, the query, key, and value projections are computed exclusively for the subset of tokens routed to this attention path.

Linear Path: If $\delta_i^{(l)} = 0$, the token undergoes a self-only linear transformation:

$$y_i^{(l)} = \text{MLP}\left(g_{i,\text{bypass}}^{(l)} \cdot x_i^{(l)} W^V W^O\right) \tag{5}$$

The projection weights $W^V, W^O \in \mathbb{R}^{d \times d}$ and $MLP(.)$ are shared across both computation paths. Tokens on the linear path strictly only compute W^V and W^O, avoiding unnecessary FLOPs. This design ensures that *all tokens are updated*, with the router dynamically choosing whether full attention is necessary.

Although routing decisions are hard at inference time, we use the soft scores $g_{i,\text{attn}}^{(l)}$ and $g_{i,\text{bypass}}^{(l)}$ to weight the outputs of both paths during training. This allows gradients to flow into the router parameters via backpropagation. This strategy is commonly used in many dynamic routing mechanisms (Shazeer et al., 2017; Fedus et al., 2022; Raposo et al., 2024). Despite the mismatch between using soft mixing for gradient flow during training and hard routing during inference, we observe high stability. The expected routing fraction closely matches the realized inference ratio ($\sim$10%), ensuring DTRNet does not suffer from the sharp performance collapse typically associated with train-test routing discrepancies.

Sparse Attention Equivalence The hard-routing scheme induces a dynamic sparse attention mechanism. Let $\tilde{g}^{(l)} = [\delta_1^{(l)}, \ldots, \delta_n^{(l)}] \in \{0,1\}^n$ denote the binary vector of routing decisions for layer l. The effective attention mask is:

$$M^{(l)} = \tilde{g}^{(l)} \cdot \tilde{g}^{(l)\top} \in \{0,1\}^{n \times n} \tag{6}$$

This mask restricts attention to interactions among tokens routed to the attention path, creating an input-dependent sparsity pattern. Unlike static sparse attention methods (e.g., Longformer (Beltagy et al., 2020), DTRNet's sparsity adapts to token importance, reducing quadratic complexity to linear for $\tilde{9}0\%$ of tokens on average (as validated in experiments). This equivalence highlights DTRNet's ability to achieve efficiency comparable to sparse attention models while maintaining flexibility through learned routing.

3.3 Training and Regularization

To encourage sparse attention usage, we train DTRNet with a composite loss. Let $g_{i,\text{attn}}^{(l)} \in [0,1]$ denote the router's soft score for sending token i to attention at layer l, and let $\delta_i^{(l)} \in \{0,1\}$ be the corresponding hard decision (1 = routed to attention). Define the per-layer *attention load*, the number of tokens not skipped as:

$$f_l^{\text{att}} = \sum_{i=1}^{n} \delta_i^{(l)},$$

where n is the sequence length. Inspired by MoE load balancing, but specialized to a single "expert" (the attention path), we penalize the product of the load and the aggregate attention scores:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \sum_{l=1}^{L} \alpha_l \left\| G^{(l)}[:, 0] \right\|_1, \quad (7)$$

where $\alpha_l = \frac{f_l^{\text{att}}}{\sum_{i=1}^{L} f_i^{\text{att}}}$ defines a layer-wise normalized attention load, to encourage balanced and efficient attention usage across layers. $\mathcal{L}_{\text{CE}}$ is the task cross-entropy loss, L is the number of layers, $\lambda > 0$ controls the strength of the routing penalty, $G^{(l)} \in \mathbb{R}^{n \times 2}$ stacks the router's per-token scores for the two paths at layer l, and $G^{(l)}[:, 0] = [g_{1,\text{attn}}^{(l)}, \ldots, g_{n,\text{attn}}^{(l)}]$. The ℓ_1 norm $\| \cdot \|_1$ aggregates attention mass across tokens. This objective discourages over-routing by increasing the penalty when many tokens are sent to attention or when their attention probabilities are large, thereby encouraging sparse, budget-aware use of attention while preserving task performance.

3.4 Architectural Design Choices

The redundancy analysis motivates two key design choices. First, the high similarity in inner layers justifies the bypass path, which uses shared projections (W^V, W^O) to update tokens at linear cost, avoiding the full skipping of layers as in MoD or D-LLM. Second, the distinct behavior of boundary layers necessitates full-attention layers at the first and last positions to handle input adaptation and output alignment. To preserve long-range dependencies, we interleave DTRNet layers with full Transformer layers (e.g., T-D-T or T-D-D-T patterns), ensuring periodic cross-token mixing. Ablation studies (see Appendix A.2) confirm that these configurations optimize the trade-off between efficiency and accuracy.

4 Experiments

Our experiments encompass a comprehensive comparison against the following recent state-of-the-art baselines: SmolLM, MoD, D-LLM. These experiments are conducted using both 360M-parameter and 1.3B-parameter model variants. To ensure fairness, all models are trained under identical conditions.

4.1 Training Setup

We consider two model scales, 360M and 1.3B parameters.

360M Models: These follow the SmolLM-360M architecture, consisting of 32 Transformer layers, a hidden dimension of 960, and an intermediate feedforward dimension of 2560. Training is done on 15B tokens randomly sampled from the FineWeb-Edu corpus (Penedo et al., 2024), using a global batch size of 384.

1.3B Models: These models consist of 24 layers, with a hidden size of 2048 and an intermediate size of 5632. They are trained on 100B tokens sampled from the FineWeb-Edu dataset (Penedo et al., 2024), with a global batch size of 1024.

All models are trained with a maximum sequence length of 2048 tokens. Optimization is performed using the AdamW optimizer (Loshchilov and Hutter, 2019) with a peak learning rate of 3e-4, weight decay of 0.01, and gradient clipping set to 0.1. The learning rate schedule follows cosine decay with a warmup ratio of 0.1. For all variants, we employ the LLaMA 2 tokenizer (Touvron et al., 2023), with a vocabulary size of 32,000.

All models are trained under a fixed FLOPs budget at 2k sequence length to ensure that observed performance differences can be attributed to architectural and routing design rather than computational disparity. Architecture and architecture-specific hyperparameters of DTRNet and other baselines are as follows:

Symmetric Tuning Protocol. To ensure a fair comparison and mitigate concerns of hyperparameter bias, we adopt a symmetric tuning protocol across all evaluated architectures. We did not perform exhaustive, task-specific hyperparameter tuning for any method. Instead, tuning was uniformly restricted to adjusting the respective routing parameters of each model to achieve a strictly matched FLOPs budget at a 2K sequence length. This ensures that observed performance differences are attributable to architectural design rather than computational or tuning disparities.

DTRNet. We explore multiple architectural configurations for integrating DTRNet blocks into Transformer models. In all variants, the first and last layers are standard Transformer layers. The main configurations are as follows:

- **DTRNet-BiLayer**: Inserts one DTRNet block after each Transformer block. (T-D-T-D-T-D)

Model	FLOPs Ratio	WIKI ppl (↓)	LMBD ppl (↓)	LMBD Acc (↑)	ARC-C Acc (↑)	ARC-E Acc (↑)	BQ Acc (↑)	HLSWG Acc (↑)	PIQA Acc (↑)	T-MMLU Acc (↑)	WG Acc (↑)	AVG Acc (↑)
360M												
SmolLM	1.00	30.18	52.59	30.93	24.32	55.72	59.88	31.80	65.72	32.62	52.88	44.23
D-LLM	0.84	32.82	63.31	30.12	22.18	54.80	60.55	31.32	63.33	31.97	48.62	42.86
MoD	0.84	32.16	50.00	31.48	22.27	54.46	61.74	31.17	64.80	27.72	51.46	43.13
DTRNet Trilayer (Ours)	0.78	31.75	64.00	29.19	23.04	55.56	61.62	31.21	66.05	31.79	51.14	43.70
DTRNet Bilayer (Ours)	0.84	30.68	56.52	31.22	23.46	56.06	59.39	31.73	66.38	32.51	54.14	**44.36**
1.3B												
SmolLM	1.00	17.44	15.61	44.67	33.28	70.24	61.96	42.02	72.96	28.93	58.17	51.53
D-LLM	0.85	18.40	17.83	43.01	31.91	68.98	60.24	40.99	71.87	29.96	58.01	50.62
MoD	0.84	19.38	18.11	42.64	33.11	68.06	56.09	41.18	71.76	29.31	55.56	49.71
DTRNet Bilayer (Ours)	0.85	17.95	16.69	43.80	32.68	69.15	60.15	41.11	71.16	32.72	57.38	**51.02**

Table 1: Comparison of DTRNet and baselines at both 360M and 1.3B scales. WIKI and LMBD are reported in perplexity (lower is better), while all other metrics are accuracy scores (higher is better). FLOPs ratio is calculated relative to the dense baseline.

- **DTRNet-TriLayer**: Inserts two DTRNet blocks after each Transformer block. (T-D-D-T-D-D)

For training, we set the regularization strength λ to 8×10^{-4} for the 360M parameter model and 6×10^{-4} for the 1.3B parameter model. To implement hard routing efficiently and dynamically skip unselected tokens from attention, we employ boolean masks in conjunction with the variable-length attention function `flash_attn_varlen_func()` from FlashAttention 2 (Dao, 2023).

MoD. We adopt the bi-layer routing configuration from the original MoD paper, placing one MoD block after each Transformer layer. We reproduce the classifier for inference with a linear layer. The top-k token routing ratio is fixed at 0.7 to maintain comparable compute (FLOPs) with other baselines for a fair evaluation.

D-LLM. For D-LLM, we set the acceleration rate $\Omega = 0.85$ and the auxiliary loss coefficient $\alpha = 1.0$, following the recommended values in the original work to have same computational budget. The router used is two layer MLP router based on original paper. The architecture retains full Transformer layers in the first two layers, while all subsequent layers are replaced with D-LLM blocks. The first two tokens of each sequence are reserved and consistently passed through the Transformer in every layer as done in original D-LLM setup. Unlike the original D-LLM setup, which applies LoRA (Hu et al., 2021) to the query/key/value projections and fine-tunes the routing components, our setup performs full model pretraining from scratch and hence does not use any parameter-efficient fine-tuning techniques.

4.2 Evaluation Benchmarks

To comprehensively evaluate DTRNet, we conduct experiments across both standard language understanding tasks and long-context extrapolation benchmarks.

Standard Language Understanding Benchmarks. To assess general language understanding, reasoning, and factual recall capabilities, we evaluate our models on a broad set of standardized benchmarks. These benchmarks span diverse tasks such as multiple-choice question answering, commonsense reasoning, and cloze-style completion. Specifically, we include evaluations on ARC (Easy and Challenge) (Clark et al., 2018), BoolQ (BQ) (Clark et al., 2019), HellaSwag (HLSWG) (Zellers et al., 2019), PIQA (Bisk et al., 2020), Tiny MMLU (T-MMLU) (Hendrycks et al., 2020), Winogrande (WG) (Sakaguchi et al., 2021), LAMBADA (LMBD) (Paperno et al., 2016) and WIKITEXT (WIKI) (Merity et al., 2016). All the evaluations on these tasks are done in zero-shot setting.

Length Extrapolation Benchmarks. In addition to standard evaluation, we assess the model's ability to extrapolate to long sequences. Following prior work, we test on sequences of up to 20K tokens using six long-context benchmarks. These tasks test long-range dependency modeling, memory retention, and contextual reasoning under extreme sequence lengths. Experiments are conducted for 1.3B models for these benchmarks. We apply YaRN (Peng et al., 2023) with a factor of 10.0 to extrapolate the sequence length. Performance on these benchmarks is reported in terms of perplexity.

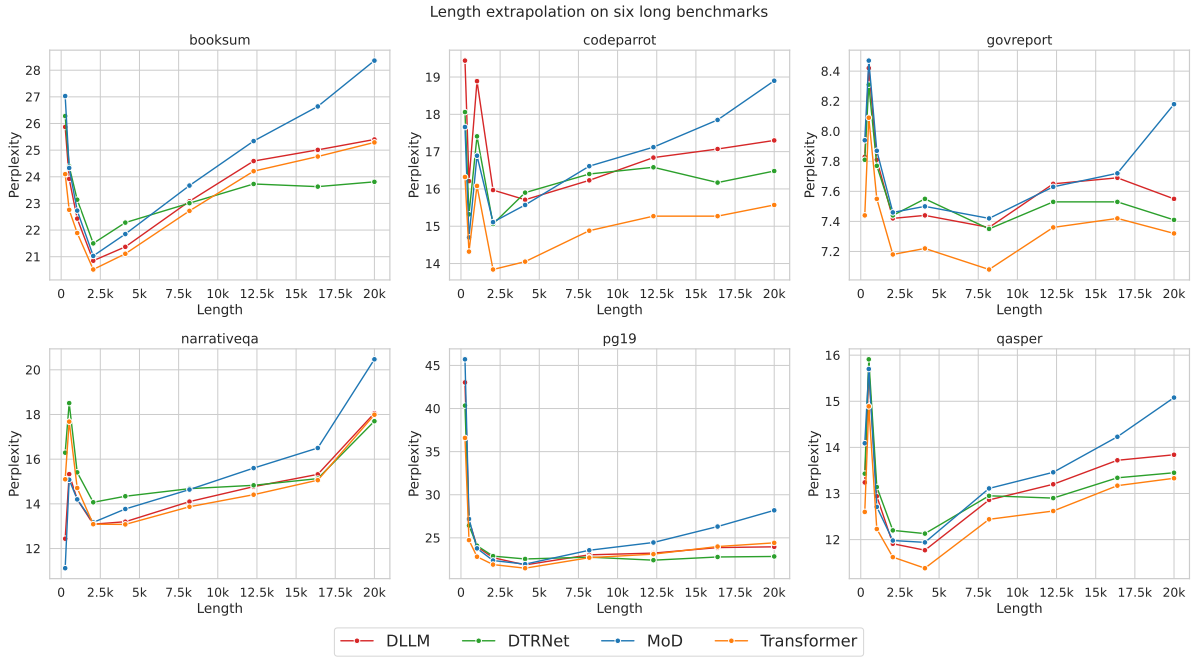


Figure 3: Perplexity across increasing sequence lengths for LongLM benchmarks on 6 different tasks. DTRNet maintains lower perplexity than MoD and D-LLM and matches or outperforms the dense Transformer on tasks like BookSum, NarrativeQA, and PG-19, demonstrating strong extrapolation to long-context inputs.

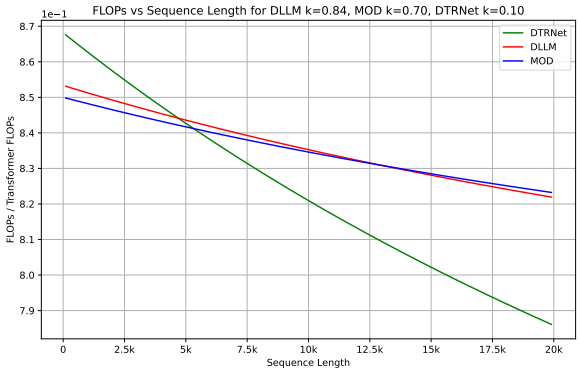


Figure 4: Theoretical FLOPs comparison as sequence length increases. DTRNet shows improved efficiency over MoD, D-LLM, and baseline Transformers for long inputs.

Memory Efficiency Analysis. In addition to task performance and extrapolation capabilities, we analyze the memory efficiency of various architectures across different sequence lengths. This analysis is crucial for understanding the practical deployment implications of each model, especially under resource-constrained settings. These analysis are performed for 360M models.

5 Results and Analysis

Standard Language Understanding Benchmarks. Table 1 presents the performance of our models on a diverse suite of standard language understanding tasks. At the 360M scale, our best-performing variant **DTRNet-BiLayer** achieves a highly competitive average accuracy (44.36), surpassing both the MoD baseline (43.13) and the D-LLM baseline (42.86). Rather than claiming absolute state-of-the-art across all individual metrics, these results demonstrate that DTRNet provides a stronger efficiency–accuracy tradeoff at matched FLOPs compared to prior routing methods. Notably, DTRNet-BiLayer performs on par with the dense Transformer baseline (SmolLM: 44.23), even exhibiting better performance in tasks such as LAMBADA, ARC-E, PIQA, and WG, indicating that accuracy can be preserved while significantly reducing compute. DTRNet-TriLayer also outperforms MoD and D-LLM on average while operating at lower FLOPs. These findings support the analysis that adjacent Transformer layers exhibit high token similarity, enabling effective token skipping across layers without substantial performance degradation.

At the 1.3B scale, DTRNet again demonstrates a strong tradeoff, achieving an average accuracy of 51.02, exceeding both MoD (49.71) and D-LLM

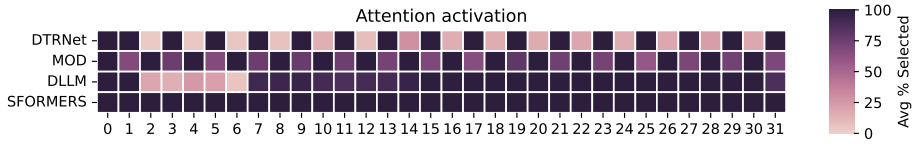


Figure 5: Average Percentage of tokens going to attention per layer for D-LLM, MoD and DTRNet. We observe much less tokens are going into attention in DTRNet while maintaining performance

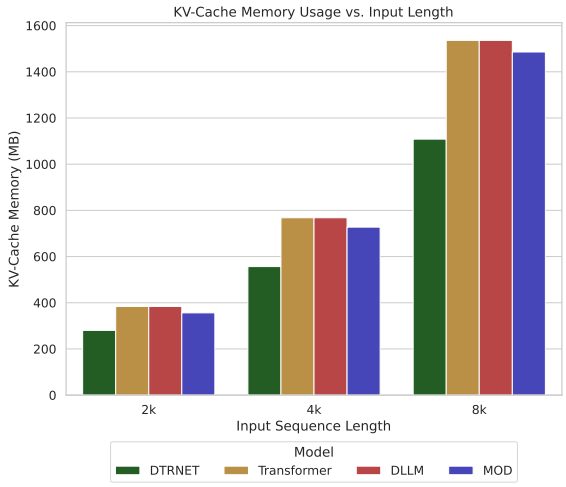


Figure 6: KV cache memory for different sequence lengths

(50.62), and closely approaching the dense Transformer baseline SmolLM (51.53). This trend is further supported by perplexity results on WIKI and LAMBADA, where DTRNet achieves lower perplexity than both sparse baselines and performs comparably to the full Transformer with reduced FLOPs. Furthermore, an ablation on the routing penalty coefficient λ (detailed in Appendix A.6) confirms that DTRNet's performance is robust across different sparsity levels and does not require delicate, task-specific hand-tuning.

Length Extrapolation Benchmarks. As shown in Figure 3, DTRNet consistently achieves lower perplexity than both MoD and D-LLM across all evaluated long-context tasks, particularly at extreme sequence lengths (e.g., 20K tokens). Notably, DTRNet matches or outperforms the dense Transformer baseline in several tasks, specifically *BookSum* (Kryściński et al., 2021), *NarrativeQA* (Kočiský et al., 2018), and *PG-19* (Rae et al., 2020), demonstrating strong generalization to longer sequences.

Importantly, these improvements come with greater computational efficiency. Figure 4 shows the relative FLOPs ratio to dense Transformer as sequence length increases. DTRNet's superiority in length extrapolation stems directly from its routing mechanism. At shorter sequence lengths, all models operate near their training distribution, resulting in minimal performance gaps. However, as sequence length increases, positional extrapolation techniques (such as YaRN) introduce out-of-distribution shifts to the attention mechanism. Because DTRNet routes significantly fewer tokens ($\sim$10%) through the quadratic attention path, it is substantially less exposed to these distributional shifts compared to MoD and D-LLM. At a 20K sequence length, DTRNet operates at a FLOPs ratio of 0.785, whereas both MoD and D-LLM maintain higher ratios around 0.82. Despite this reduced computational cost, DTRNet maintains strong generation quality, highlighting its effective token selection strategy.

Memory Efficiency Analysis. Figure 5 illustrates the percentage of tokens routed to the attention mechanism during inference at each layer for the DTRNet and baseline models. DTRNet passes only 10% tokens to attention per layer, and this selection remains remarkably uniform across all DTRNet layers. This consistent sparsity is a direct result of the weighted regularization introduced during training, which encourages balanced token flow across layers and prevents overreliance on specific depths. In contrast, D-LLM lacks an explicit load balancing mechanism, leading to an imbalanced token distribution. As seen in the figure, the initial D-LLM layers route very few tokens to attention, an effect also observed during training, which results in starvation of these layers and subsequently degrades overall performance. MoD, on the other hand, maintains a more uniform token flow, with approximately 70% of tokens being routed through attention across layers. However, this behavior is driven by expert choice routing during training and replicated at inference time using a learned classifier. This indirect mechanism limits MoD's ability

to adapt to dynamic token importance, potentially impacting its effectiveness.

Passing fewer tokens to attention not only reduces computational cost but also significantly lowers memory usage, particularly in the context of key-value (KV) caching during autoregressive inference. As shown in Figure 6, DTRNet consistently consumes substantially less memory for KV cache compared to other baselines under matched FLOPs budgets. This advantage becomes increasingly noticeable as the sequence length grows, highlighting DTRNet's suitability for long-context inference. While D-LLM proposes a KV eviction strategy to mitigate memory usage, its implementation involves masking the KV cache during attention computation. Consequently, this method does not reduce the actual KV cache footprint, and as observed in Figure 6, its memory usage remains comparable to that of a dense Transformer. In contrast, DTRNet achieves true memory savings by avoiding KV allocation for unselected tokens entirely, making it a more practical and scalable solution for memory-efficient inference. These results underscore the effectiveness of DTRNet in balancing compute and performance, offering strong length extrapolation capabilities while maintaining or improving performance with reduced quadratic attention overhead.

6 Conclusion

We introduced DTRNet, a dynamic token routing architecture that reduces Transformer inference cost by decoupling attention from token updates. By replacing full-layer skipping with a linear path that retains the MLP, DTRNet ensures that all tokens receive meaningful updates, even when attention is bypassed. This lightweight linear path enables substantial compute savings while maintaining accuracy. Through extensive experiments, we showed that DTRNet outperforms SOTA approaches MoD and D-LLM in both accuracy and memory at matched FLOPs, particularly on long-context tasks. Our work highlights the utility of projection-based updates and fine-grained token routing in building efficient, high-performance language models, paving the way for scalable deployment of LLMs in latency-sensitive and low memory settings.

7 Limitations

Despite the strong performance of DTRNet, the introduction of a routing mechanism—similar in spirit to mixture-of-experts models—adds modest complexity during training. The router is lightweight and introduces limited additional hyperparameters, with the associated overhead confined to the training phase; inference remains simple and efficient once the routing policy is learned.

The current routing design relies on heuristic signals that are stable and effective across the evaluated models and tasks. However, these heuristics may not be optimal in all settings, particularly for substantially larger models or workloads with different computational profiles. Developing more adaptive or learned routing strategies is a natural direction for future work.

Finally, our evaluation focuses on decoder-only language models and standard NLP benchmarks. While the proposed architecture is not inherently restricted to this regime, extending DTRNet to encoder–decoder architectures, multimodal models, and more challenging reasoning workloads remains an open area for further investigation.

References

Anthropic. 2023. Claude: A next-generation ai assistant by anthropic. https://www.anthropic.com/index/introducing-claude. Accessed: 2025-07-28.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. 2024. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. 2024. Flextron: Many-in-one flexible large language model. *arXiv preprint arXiv:2406.10260*.

Zouying Cao, Yifei Yang, and Hai Zhao. 2024. Headwise shareable attention for large language models. *arXiv preprint arXiv:2402.11819*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations (ICLR)*.

Yihe Dong, Lorenzo Noci, Mikhail Khodak, and Mufan Li. 2025. Attention retrieves, mlp memorizes: Disentangling trainable components in the transformer. *arXiv preprint arXiv:2506.01115*.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, and 1 others. 2024. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270.

Google Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv: 2403.05530*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

J. E. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*.

Benyamin Jamialahmadi, Parsa Kavehzadeh, Mehdi Rezagholizadeh, Parsa Farinneya, Hossein Rajabzadeh, Aref Jafari, Boxing Chen, and Marzieh S Tahaei. 2025. Balcony: A lightweight approach to dynamic inference of generative language models. *arXiv preprint arXiv:2503.05005*.

Yikun Jiang, Huanyu Wang, Lei Xie, Hanbin Zhao, Chao Zhang, Hui Qian, and John C.S. Lui. 2024. D-llm: A token adaptive computing resource allocation strategy for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 1725–1749. Curran Associates, Inc.

Parsa Kavehzadeh, Mojtaba Valipour, Marzieh Tahaei, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. 2024. Sorted llama: Unlocking the potential of intermediate layers of large language models for dynamic inference. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2129–2145.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2021. Booksum: A collection of datasets for long-form narrative summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*. ArXiv:1711.05101.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

OpenAI. 2023. Gpt-4 technical report. *PREPRINT*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

Bowen Peng, Jeffery Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context extension for large language models. *arXiv preprint arXiv:2309.00071*.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. 2024. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Noam M. Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and J. Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations*.

Seungjun Shin, Jaehoon Oh, and Dokwan Oh. 2025. Orthorank: Token selection via sink token orthogonality for efficient llm inference. *Preprint*, arXiv:2507.03865.

Hugo Touvron, Louis Martin, Kevin Stone, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Hossein Rajabzadeh, Marzieh S Tahaei, Boxing Chen, and Ali Ghodsi. 2023. Sortednet, a place for every network and every network in its place: Towards a generalized solution for training many-in-one neural networks. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Haoyi Wu and Kewei Tu. 2024. Layer-condensed kv cache for efficient inference of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11175–11188.

Zeping Yu and Sophia Ananiadou. 2023. Neuron-level knowledge attribution in large language models. *arXiv preprint arXiv:2312.12141*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, and 1 others. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114.

A Appendix

A.1 Expert Choice vs Token Choice Routing

Benchmark	Smollm 360M	DTRNet Expert Choice	DTRNet Token Choice
FLOPs	1.00	0.87	0.85
ARC-C Acc (↑)	24.32	23.81	23.46
ARC-E Acc (↑)	55.72	55.98	56.06
BQ Acc (↑)	59.88	57.00	59.39
HLSWG Acc (↑)	31.80	31.66	31.73
LMBD Acc (↑)	30.93	30.71	31.22
PIQA Acc (↑)	65.72	65.61	66.38
T-MMLU Acc (↑)	32.62	26.80	32.51
WG Acc (↑)	52.88	51.30	54.14
AVG Acc (↑)	44.23	42.85	44.36

Table 2: Performance of DTRNet (360M) with token-choice and expert-choice routing strategies.

Expert Choice routing assigns tokens to different experts based on a global decision computed over the entire input sequence. This approach works well during training, where all tokens are available at once. However, during autoregressive inference, tokens arrive sequentially and the model does not have access to the full sequence at once. This mismatch between training and inference introduces inconsistency in the routing behavior, leading to performance degradation.

Token Choice routing, on the other hand, makes routing decisions independently for each token. This matches the inference-time scenario, ensuring consistent behavior across training and inference. To empirically compare the two strategies, we implemented an Expert Choice variant of DTRNet where the top 25% of tokens are routed through attention, while the rest are skipped. The model architecture remained identical to the Token Choice setup, differing only in the routing mechanism. During training, both variants exhibited similar training losses. However, as shown in Table 2, the Token Choice model consistently outperforms its Expert Choice counterpart across several benchmarks and achieves a higher overall average accuracy.

Given its superior alignment with autoregressive inference and stronger empirical results, we adopt Token Choice routing as the routing mechanism in all final DTRNet models.

A.2 DTRNet Architecture Ablations

We explore multiple architectural configurations for integrating DTRNet blocks into Transformer models. In all variants, the first and last layers are standard Transformer layers. The main configurations are as follows:

Dataset / Metric	DTRNet Trilayer	DTRNet Laterhalf	DTRNet 6 Transformer	DTRNet Bilayer
WIKI ppl (↓)	31.75	31.25	30.78	30.68
LMBD ppl (↓)	64.00	63.96	57.77	56.52
LMBD Acc (↑)	29.19	29.23	29.81	31.22
ARC-C Acc (↑)	23.04	22.87	23.29	23.46
ARC-E Acc (↑)	55.56	54.71	57.11	56.06
BQ Acc (↑)	61.62	57.46	58.87	59.39
HLSWG Acc (↑)	31.21	31.36	31.75	31.73
PIQA Acc (↑)	66.05	65.45	65.07	66.38
T-MMLU Acc (↑)	31.79	30.34	28.13	32.51
WG Acc (↑)	51.14	51.93	51.38	54.14
AVG Acc (↑)	43.70	42.92	43.18	44.36

Table 3: Comparison of four DTRNet architectural variants (360M)

- **DTRNet-BiLayer**: Inserts one DTRNet block after each Transformer block. (T-D-T-D-...-D-T)
- **DTRNet-TriLayer**: Inserts two DTRNet blocks after each Transformer block. (T-D-D-T-D-D-...-T)
- **DTRNet-LaterHalf**: The first half of the model consists of full Transformer layers, while the second half consists of DTRNet blocks. (T-T-T-...-D-D-...-D)
- **DTRNet-6Transformer**: A hybrid design where the first two, middle two, and final two layers are full Transformers (6 layers in total), and the remaining layers are DTRNet.

Table 3 compares four DTRNet architectural configurations on standard language understanding tasks. Among these, DTRNet-BiLayer achieves the highest average accuracy (44.36), followed closely by DTRNet-TriLayer (43.70), outperforming both the LaterHalf (42.92) and 6-Transformer (43.18) variants. This trend is consistent across both perplexity and accuracy metrics, where BiLayer achieves the best performance on most benchmarks, including WIKI, LMBD, ARC-C, BoolQ, PIQA, and Winogrande.

These results support our analysis: token representations in adjacent Transformer layers exhibit high cosine similarity. As a result, selectively skipping attention computation for similar tokens across neighboring layers introduces minimal degradation in contextual understanding. Architectures like BiLayer and TriLayer, which interleave DTRNet blocks with dense layers, exploit this redundancy effectively, preserving performance

while reducing computational overhead. In contrast, LaterHalf and 6-Transformer configurations separate DTRNet blocks more coarsely or cluster dense layers, making it harder to leverage local token redundancy, thereby leading to lower performance. This confirms that fine-grained sparsity across adjacent layers offers better performance.

A.3 Effect of Skipping All Attention

Dataset / Metric	SmolLM 360M	DTRNet Bilayer	DTRNet Skip
WIKI ppl (↓)	30.18	30.68	30.95
LMBD ppl (↓)	52.59	56.52	61.16
LMBD Acc (↑)	30.93	31.22	28.97
ARC-C Acc (↑)	24.32	23.46	23.04
ARC-E Acc (↑)	55.72	56.06	56.27
BQ Acc (↑)	59.88	59.39	54.10
HLSWG Acc (↑)	31.80	31.73	31.28
PIQA Acc (↑)	65.72	66.38	64.80
T-MMLU Acc (↑)	32.62	32.51	31.76
WG Acc (↑)	52.88	54.14	49.80
FLOPs Ratio	1.00	0.85	0.81
AVG Acc (↑)	44.23	44.36	42.50

Table 4: Comparison of SmolLM, DTRNet Bilayer and DTRNet Skip (360M)

To assess the necessity of routing tokens through the quadratic attention path, we introduce a variant named DTRNet-Skip, where no tokens are passed to the standard multi-head attention in any DTRNet layer. Instead, all tokens are directly forwarded through the linear path. Importantly, the architecture remains identical to the DTRNet-BiLayer configuration, alternating Transformer and DTRNet blocks, with the only difference being that the DTRNet routers are configured to skip all tokens from attention.

As shown in Table 4, although DTRNet-Skip reduces FLOPs by bypassing the most expensive quadratic operation entirely, this comes at a clear cost to performance. The average accuracy drops from 44.36 (DTRNet-BiLayer) to 42.50, and the model underperforms across most tasks. Furthermore, perplexity increases on both WIKI and LAMBADA.

These results confirm that completely avoiding attention degrades model quality. While MLP paths can process and transform information, they lack the global context modeling provided by attention. This experiment demonstrates that selectively passing a small fraction of important tokens to the attention mechanism is essential for maintaining competitive performance while still achieving computational efficiency.

A.4 MoD and D-LLM original variants

Dataset / Metric	D-LLM (0.55)	D-LLM (0.85)	MoD (k=0.125)	MoD (k=0.7)	DTRNet Bilayer
WIKI ppl (↓)	35.13	32.82	33.80	32.16	30.68
LMBD ppl (↓)	83.61	63.31	63.10	50.00	56.52
LMBD Acc (↑)	27.13	30.12	28.97	31.48	31.22
ARC-C Acc (↑)	22.70	22.18	21.50	22.27	23.46
ARC-E Acc (↑)	52.27	54.80	51.98	54.46	56.06
BQ Acc (↑)	60.12	60.55	57.77	61.74	59.39
HLSWG Acc (↑)	30.42	31.32	30.95	31.17	31.73
PIQA Acc (↑)	64.47	63.33	63.82	64.80	66.38
T-MMLU Acc (↑)	30.79	31.97	31.76	27.72	32.51
WG Acc (↑)	50.59	48.62	50.20	51.46	54.14
FLOPs Ratio	0.56	0.85	0.55	0.85	0.85
AVG Acc (↑)	42.31	42.86	42.12	43.14	44.36

Table 5: Comparison of original MoD, D-LLM variants with our increased FLOPs variants, and DTRNet Token Choice (360M)

To ensure a fair comparison with DTRNet, we also reimplemented and evaluated the best-performing configurations of MoD and D-LLM as reported in their original papers ($k = 0.125$ for MoD, and accelaration rate 0.55 for D-LLM), using identical model size (360M), training data, and evaluation protocol. These original variants operate at lower FLOPs ratios because they route fewer tokens through the Transformer layer.

As shown in Table 5, while these variants are efficient, they suffer from reduced performance across most benchmarks compared to their higher FLOPs counterparts. This degradation highlights a fundamental trade-off in sparse routing: reducing computation by skipping too many tokens can hurt model quality. To allow for a more fair comparison, in our main results (Table 1), we increased the FLOPs budget for both MoD and D-LLM by routing more tokens ($k = 0.7$ for MoD, and accelaration rate 0.85 for D-LLM). These higher FLOPs versions show improved performance, confirming that performance improves with increased token routing. However, even under matched compute, DTRNet consistently outperforms both MoD and D-LLM, demonstrating the effectiveness of our learned token selection strategy.

A.5 Value and Output Projections Are Needed for Skipped Tokens

In DTRNet, tokens routed to the bypass path skip the quadratic attention computation but are still passed through the value (W^V) and output (W^O) projections before the MLP. This operation is equivalent to a form of *self-attention without interaction*,

Model	WIKI ppl (↓)	LMBD ppl (↓)	ARC-C Acc (↑)	HLSWG Acc (↑)	PIQA Acc (↑)	T-MMLU Acc (↑)	WG Acc (↑)	AVG Acc (↑)	FLOPs Ratio
SmolLM (baseline)	30.18	52.59	24.32	31.80	65.72	32.62	52.88	42.79	1.00
DTRNet ($\lambda = 0.001$)	31.49	63.28	23.46	31.70	65.13	29.09	53.51	42.45	0.87
DTRNet ($\lambda = 0.0008$)	30.68	56.52	23.46	31.73	66.38	32.51	54.14	42.86	0.85
DTRNet ($\lambda = 0.0005$)	30.37	54.22	24.66	31.98	64.53	30.21	50.67	42.14	0.81

Table 6: Ablation of the routing penalty coefficient λ on the 360M scale model. Accuracy metrics are reported in percentages.

Dataset / Metric	DTRNet (Bilayer)	DTRNet (Bilayer w/o vo)
WIKI ppl (↓)	30.68	32.28
LMBD ppl (↓)	56.52	49.97
LMBD Acc (↑)	31.22	30.97
ARC-C Acc (↑)	23.46	22.70
ARC-E Acc (↑)	56.06	55.35
BQ Acc (↑)	59.39	50.55
HLSWG Acc (↑)	31.73	31.50
PIQA Acc (↑)	66.38	64.47
T-MMLU Acc (↑)	32.51	28.67
WG Acc (↑)	54.14	51.78
AVG Acc (↑)	44.36	41.99

Table 7: Comparison of DTRNet Bilayer with and without value and output projections under matched hyperparameters (360M)

where a token attends only to itself. By applying $W^V W^O$, the token's representation is transformed in the same space as tokens from the attention path, maintaining consistency across both routes.

Table 7 shows that removing these projections leads to a clear performance drop across benchmarks (average accuracy falling from 44.36 to 41.99). This confirms that even for skipped tokens, passing them through W^V and W^O enables richer updates and better representational alignment, resulting in improved overall performance compared to simply forwarding them unchanged to the MLP.

A.6 Sensitivity Analysis of Routing Penalty (λ)

To investigate the effect of the routing penalty coefficient λ and address potential concerns regarding hyperparameter sensitivity, we conducted an ablation study at the 360M scale. We varied λ within the range $[0.0005, 0.001]$, which corresponds to models operating at 81% to 87% of the full-attention FLOPs budget. All models were trained under the identical setup described in Section 4.1.

As shown in Table 6, the average downstream accuracy remains highly stable (ranging from 42.14 to 42.86) despite meaningful differences in computational cost. Importantly, no sharp performance collapse is observed as sparsity increases. This indicates that DTRNet is not highly sensitive to precise λ selection. Rather, λ serves as a smooth control knob for attention sparsity, allowing the architecture to operate efficiently along the efficiency-accuracy frontier without the need for delicate, task-specific hand-tuning.