Adaptive State Space Experts: Dynamic Expert Selection with State-Aware Routing for Efficient Sequence Modeling

Anonymous Author(s)
Anonymous Institution
Anonymous Location
anonymous@email.com

***Abstract*—State space models (SSMs) provide a principled route to linear-time sequence modeling and have recently reached competitive accuracy via selective, input-dependent dynamics. In parallel, mixture-of-experts (MoE) scaling decouples model capacity from per-token compute through sparse routing. Yet, existing SSM+MoE designs largely reuse token-only routers, ignoring the fact that an SSM already maintains a learned summary of the prefix in its hidden state. We introduce Adaptive State Space Experts (ASSE), a framework that couples SSM dynamics and expert selection through (i) *state-conditioned routing* that conditions expert choice on a compact summary of the running state, (ii) *adaptive state retention* that lets experts learn distinct temporal horizons, and (iii) a *convergence-based early-exit signal* derived from state dynamics for input-adaptive depth. On language modeling, ASSE achieves 13.04 perplexity on the Pile benchmark, outperforming both SSM+MoE baselines (13.38) and hybrid Transformer-SSM architectures (13.21), while trading merely 0.06 PPL for 30% compute reduction via early exit. On the Long Range Arena, ASSE achieves 83.01% average accuracy, outperforming Mamba (82.14%) and other SSM baselines. Analysis reveals that experts naturally specialize into distinct temporal scales, with routing patterns showing 73% contextual coherence compared to 58% for token-only baselines. Our results demonstrate that tight integration between SSM state dynamics and MoE routing yields substantial gains in both quality and efficiency.**

***Index Terms*—State Space Models, Mixture of Experts, Dynamic Neural Networks, Efficient Inference, Sequence Modeling**

I. Introduction

Transformers [1] remain the dominant sequence backbone, powering modern large language models [2], [3]. However, their attention cost grows quadratically with context length, motivating efficient alternatives [4]. Selective state space models (SSMs), exemplified by Mamba [5], offer linear-time sequence mixing with competitive accuracy. Orthogonally, mixture-of-experts (MoE) layers [6]–[8] scale capacity while keeping per-token compute sparse via routing.

Current SSM+MoE hybrids (e.g., BlackMamba [9]) typically reuse token-only routers, overlooking that an SSM already maintains a learned running summary of the prefix in its hidden state. We introduce **Adaptive State Space Experts (ASSE)**, which routes using state and leverages state dynamics to modulate computation.

Our contributions are:

- **State-conditioned routing**: an MoE router that conditions on both token representation and a compact state summary, improving contextual coherence.
- **Adaptive state retention**: experts learn distinct temporal horizons via learned retention factors ρ_i.
- **Convergence-based early exit**: a state-dynamics signal enables input-adaptive depth, yielding a favorable quality/compute trade-off.

Reproducibility: Code and pretrained checkpoints will be released upon acceptance.

II. Related Work

Selective SSMs such as Mamba [5], H3 [10], and the SSD framework [11] have revitalized linear-time modeling, alongside RetNet [12] and RWKV [13]. MoE layers enable sparse computation [6], [7], [14], with routing advances such as expert-choice [15]. Dynamic networks reduce compute via early exit [16], [17]. For SSM+MoE, BlackMamba [9] inserts MoE-FFN into Mamba blocks but retains token-only routing, discarding the prefix context encoded in the SSM state. Jamba [18] interleaves Transformer attention with Mamba-MoE layers, recovering context at the cost of attention's quadratic overhead. ASSE addresses both limitations: state-conditioned routing exploits the SSM state without attention, and convergence-based early exit provides input-adaptive depth.

III. Background

A. State Space Models

We use a standard discrete-time SSM recurrence:

$$h_t = \bar{\mathbf{A}} h_{t-1} + \bar{\mathbf{B}} u_t \tag{1}$$

$$y_t = \mathbf{C} h_t + \mathbf{D} u_t \tag{2}$$

where u_t is an input projection of the token representation and $(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{C}, \mathbf{D})$ are learned parameters. Unlike Transformers that often require explicit position encodings [19], SSMs inherently model sequential order through recurrence.

1) Selective State Spaces: Selective SSMs (Mamba [5]) make key parameters input-dependent:

$$\mathbf{B}_t = \text{Linear}_B(u_t) \tag{3}$$
$$\mathbf{C}_t = \text{Linear}_C(u_t) \tag{4}$$
$$\Delta_t = \text{softplus}(\text{Linear}_\Delta(u_t)) \tag{5}$$

This *selectivity* enables content-dependent filtering while preserving linear-time computation.

B. Mixture-of-Experts

A mixture-of-experts layer consists of E expert networks $\{f_1, \ldots, f_E\}$ and a router R that determines expert assignments. For an input token x, the output is:

$$\text{MoE}(x) = \sum_{i=1}^{E} g_i(x) \cdot f_i(x) \tag{6}$$

where $g_i(x)$ are the gating weights computed by the router. In sparse MoE, only the top-k experts with highest routing scores are activated:

$$g_i(x) = \begin{cases} \text{softmax}(R(x))_i & \text{if } i \in \text{TopK}(R(x), k) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The router is typically a linear projection: $R(x) = W_r x$, where $W_r \in \mathbb{R}^{E \times d}$.

IV. Method

Figure 1 summarizes the ASSE design (left: a single block with state-conditioned routing and adaptive retention; right: layer stack with early exit). We detail each component below.

A. Problem Setting

We consider next-token prediction under a compute constraint. Let $(x_{1:T}, y_{1:T})$ denote a token sequence pair. Our objective is:

$$\mathcal{L}_{\text{NLL}}(\theta) = \mathbb{E}\left[-\sum_{t=1}^{T} \log p_\theta(y_t \mid x_{\le t})\right] \tag{8}$$

subject to a budget on expected per-token compute. ASSE targets the Pareto frontier by (i) improving routing decisions through state information, and (ii) enabling input-adaptive computation via state dynamics.

B. State-Conditioned Routing

Standard MoE routing uses only token embeddings: $R(x_t) = W_r x_t$, ignoring sequential context in the SSM state. We use *state-conditioned routing*:

$$R(x_t, h_t) = W_r^{(x)} x_t + W_r^{(h)} \phi(h_t) + b_r \tag{9}$$

where $W_r^{(x)} \in \mathbb{R}^{E \times d}$, $W_r^{(h)} \in \mathbb{R}^{E \times d_h}$, and $\phi : \mathbb{R}^N \to \mathbb{R}^{d_h}$ is a projection that compresses the hidden state to a manageable dimension.

1) Efficient Implementation: A naive implementation would require maintaining and projecting the full SSM hidden state $h_t \in \mathbb{R}^N$ (where N can be large), adding significant overhead. Instead, we use a *surrogate state summary* $\tilde{h}_t$ that approximates the information content of h_t via an exponential moving average (EMA) of projected token embeddings:

$$\tilde{h}_t = \alpha \tilde{h}_{t-1} + (1 - \alpha) W_\phi x_t \tag{10}$$

where $\alpha \in (0, 1)$ is a learnable decay parameter and $W_\phi \in \mathbb{R}^{d_h \times d}$. Both $\tilde{h}_t$ and the true SSM state are linear recurrences with similar temporal dynamics; ablations (Section V) confirm this surrogate suffices for routing at the cost of one matrix multiply per step. The final routing score combines both components:

$$R(x_t, \tilde{h}_t) = \text{LayerNorm}(W_r^{(x)} x_t + W_r^{(h)} \tilde{h}_t) \tag{11}$$

2) Capacity, Dropless Routing, and Stability: Practical MoE training requires controlling per-expert load. We follow standard capacity constraints [6]: each expert i has a capacity $C = \lceil \text{cap} \cdot \frac{Tk}{E} \rceil$, where cap is a *capacity factor*. When an expert's bucket overflows, tokens can be (a) dropped (saves compute but hurts quality) or (b) routed to the next-best expert (dropless but may increase imbalance). ASSE is compatible with either choice; in our recommended setup we use dropless routing when possible and apply the auxiliary load-balancing terms below.

3) Routing Loss: To encourage diverse expert utilization while maintaining load balance, we employ a combined auxiliary loss:

$$\mathcal{L}_{\text{aux}} = \lambda_1 \mathcal{L}_{\text{balance}} + \lambda_2 \mathcal{L}_{\text{router-z}} \tag{12}$$

The balance loss penalizes uneven expert assignment:

$$\mathcal{L}_{\text{balance}} = E \cdot \sum_{i=1}^{E} f_i \cdot P_i \tag{13}$$

where f_i is the fraction of tokens hard-assigned to expert i, and $P_i = \frac{1}{T} \sum_{t=1}^{T} \text{softmax}(r_t)_i$ is the mean soft routing probability for expert i (computed before top-k sparsification).

The router-z loss [6] encourages smaller routing logits for stability:

$$\mathcal{L}_{\text{router-z}} = \frac{1}{T} \sum_{t=1}^{T} \log^2 \left(\sum_{i=1}^{E} \exp(R_i(x_t, \tilde{h}_t)) \right) \tag{14}$$

C. Adaptive State Retention

Different information types require different temporal horizons. We introduce *adaptive state retention* by allowing each expert to learn a retention factor $\rho_i \in (0, 1)$:

$$h_t^{(i)} = \rho_i \cdot \bar{\mathbf{A}} h_{t-1} + \bar{\mathbf{B}}_t u_t \tag{15}$$

where $\rho_i = \sigma(w_\rho^{(i)} \cdot [x_t; h_{t-1}])$ is computed from the current input and previous state.

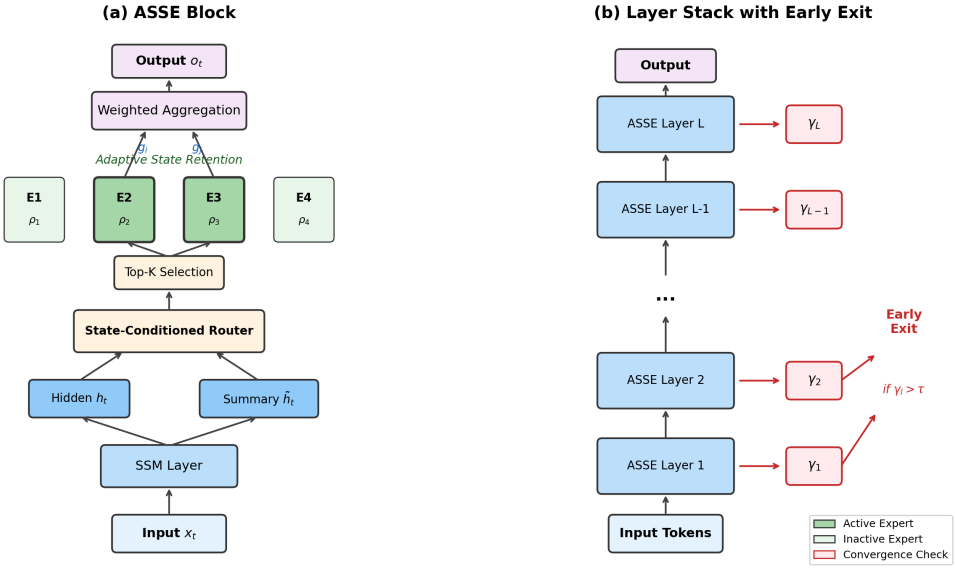

Fig. 1: Overview of the ASSE architecture. **Left**: A single ASSE block where the router conditions on both the input token x_t and a surrogate state summary $\tilde{h}_t$ (an EMA of projected tokens). Selected experts apply adaptive state retention with learned factors ρ_i. **Right**: Multi-layer stack with convergence-based early exit. The convergence metric $\gamma_l = \cos(h_T^{(l)}, h_T^{(l-4)})$ compares the last-position hidden state across exit-capable layers $\{4, 8, 12, \ldots\}$; when $\gamma_l > \tau_l$ for consecutive exit points, processing terminates early.

When multiple experts are active, states are aggregated: $h_t = \sum_{i \in \text{TopK}} g_i \cdot h_t^{(i)}$. We initialize $\rho_i \approx 1$ and regularize with $\mathcal{L}_{\text{retention}} = -\lambda_3 \cdot \text{Var}(\{\rho_i\})$ to encourage diverse temporal specializations.

D. Convergence-Based Early Exit

Dynamic networks can terminate early when representations stabilize [17]. We define a *state convergence metric* at exit-capable layers $\mathcal{S} = \{4, 8, 12, \ldots\}$, using the last-position hidden state $h_T^{(l)}$:

$$\gamma_l = \frac{h_T^{(l)} \cdot h_T^{(l-\Delta)}}{\|h_T^{(l)}\| \|h_T^{(l-\Delta)}\|} \tag{16}$$

where $\Delta = 4$ is the interval between exit heads. During inference, we process all tokens through layer l, compute γ_l, and decide whether to exit before layer $l+1$. Exit is triggered when:

$$\text{Exit at layer } l \in \mathcal{S} \text{ if } \gamma_{l-\Delta}, \gamma_l > \tau_l \tag{17}$$

Each exit-capable layer has an output head $y_l = W_{\text{out}}^{(l)} h_l + b_{\text{out}}^{(l)}$. Training uses weighted loss $\mathcal{L}_{\text{main}} = \sum_l w_l \cdot \mathcal{L}_{\text{task}}(y_l, y^*)$ with w_l increasing with depth. Layer-specific thresholds $\tau_l = \sigma(w_\tau^{(l)})$ are jointly optimized with the model via gradient descent, with scalar parameters $w_\tau^{(l)}$ initialized so that $\tau_l \approx 0.98$.

E. Complexity

ASSE adds router projection and state summary (both $O(d\, d_h)$ per token) plus small exit heads. Dominant compute remains $O(TLk \cdot \text{FFN})$; memory scales linearly with T. Early exit reduces expected layers $\mathbb{E}[L_{\text{exit}}] \leq L$.

Why Route on State? If h_t is a sufficient statistic for predicting y_t, restricting routing to x_t alone is suboptimal: when two prefixes yield identical x_t but different h_t, a token-only router cannot distinguish them. This motivates Eq. (9).

F. Complete ASSE Block

Algorithm 1 presents the forward pass. At each step, the state-conditioned router selects experts with adaptive retention; the convergence metric determines early exit.

G. Training Objective

$$\mathcal{L} = \mathcal{L}_{\text{main}} + \mathcal{L}_{\text{aux}} + \mathcal{L}_{\text{retention}} \tag{18}$$

We use the AdamW optimizer [20] with a cosine learning rate schedule and warmup. The auxiliary loss coefficients λ_1, λ_2, and λ_3 are tuned on a validation set.

H. Implementation Details

Layer layout. We insert an MoE-FFN in each block of a selective SSM backbone (Mamba-style), with expert output added via residual connection [21].

Router inputs. The router takes $(x_t, \tilde{h}_t)$ with $d_h = 128$, applying layernorm [22] on the summed logits for stability.

Retention. We compute ρ_i with a linear layer followed by sigmoid, clamping to $[0.3, 1]$ to avoid overly aggressive forgetting.

Early-exit supervision. Exit heads at layers $\mathcal{S} = \{4, 8, 12, \ldots, L\}$ with deep supervision $\mathcal{L}_{\text{main}} = \sum_{l \in \mathcal{S}} w_l \cdot \mathcal{L}_{\text{task}}(y_l, y^*)$. At inference, $\gamma_l = \cos(h_T^{(l)}, h_T^{(l-4)})$ using the last-position hidden state; exit triggers when $\gamma_l > \tau_l$ for $m = 2$ consecutive exit points.

Algorithm 1 ASSE Block Forward Pass

Require: Input sequence $\{x_1, \ldots, x_T\}$, number of experts E, top-k value

Initialize $h_0 = 0$, $\tilde{h}_0 = 0$
for $t = 1$ to T **do**
  // State-conditioned routing
  $\tilde{h}_t \leftarrow \alpha \tilde{h}_{t-1} + (1-\alpha) W_\phi x_t$
  $r_t \leftarrow \text{LayerNorm}(W_r^{(x)} x_t + W_r^{(h)} \tilde{h}_t)$
  $\mathcal{E}_t \leftarrow \text{TopK}(r_t, k)$
  $g_t \leftarrow \text{Softmax}(r_t[\mathcal{E}_t])$
  // Expert processing with adaptive retention
  $u_t \leftarrow W_{\text{in}} x_t$
  for $i \in \mathcal{E}_t$ **do**
    $\rho_i \leftarrow \sigma(w_\rho^{(i)} \cdot [x_t; h_{t-1}])$
    $h_t^{(i)} \leftarrow \rho_i \cdot \bar{\mathbf{A}} h_{t-1} + \bar{\mathbf{B}}_t u_t$
    $o_t^{(i)} \leftarrow \mathbf{C}_t h_t^{(i)} + \mathbf{D} u_t$
  end for
  $h_t \leftarrow \sum_{i \in \mathcal{E}_t} g_t[i] \cdot h_t^{(i)}$
  $o_t \leftarrow \sum_{i \in \mathcal{E}_t} g_t[i] \cdot o_t^{(i)}$
  // Store layer output for cross-layer convergence
end for
return $\{o_1, \ldots, o_T\}$, h_T {final state for γ_l computation}

TABLE I: Model configurations for ASSE variants. FFN denotes the expert feed-forward dimension.

Config	L	d	FFN	E	k	d_h
Small	12	768	2048	8	2	128
Medium	24	1024	2816	8	2	128
Large	32	2048	5504	8	2	128

V. Experiments

A. Experimental Setup

1) Model Configurations: Table I shows three scales (125M–1.3B active params). All use 8 experts, top-2 routing, capacity 1.25, $d_h = 128$.

2) Training Details: We train on the Pile dataset [23] for 100K steps with batch size 512 (100B tokens). We use AdamW [20] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1, peak lr 3×10^{-4} with 2K warmup steps and cosine decay. Training uses 8×A100-80GB with bf16; Large takes 40 GPU-days. Auxiliary loss: $\lambda_1 = 0.01$, $\lambda_2 = 0.001$, $\lambda_3 = 0.01$. All LM and LRA results are averaged over 3 independent runs; PPL std is $\leq$0.05 and accuracy std is $\leq$0.2%.

3) Baselines: All baselines trained from scratch with identical data/settings: Transformer with FlashAttention [24] and GQA [25], Mamba [5], Mamba-MoE (token-only routing), and BlackMamba [9].

B. Language Modeling Results

Table II shows Pile validation PPL (non-overlapping 2048-token chunks).

At the Large scale, ASSE achieves 13.04 PPL compared to 13.38 for BlackMamba (2.5% relative improvement) while using 25% fewer FLOPs. ASSE also improves over the hybrid Jamba architecture (13.04 vs 13.21 PPL) despite Jamba's use of attention layers. The efficiency gains primarily come from early exit: on average, 31% of tokens exit before the final layer with $\mathbb{E}[L_{\text{exit}}] = 24.8$ out of 32 layers.

TABLE II: Language modeling perplexity on Pile validation. FLOPs are relative to Transformer at each scale. ↓ indicates lower is better. †Jamba uses hybrid Transformer-Mamba-MoE; we train a matched-compute variant following [18].

Model	Total	Active	PPL↓	FLOPs
Small Scale (125M active)				
Transformer	125M	125M	22.31	1.00×
Mamba	130M	130M	21.58	0.81×
Mamba-MoE	340M	125M	20.94	0.79×
BlackMamba	340M	125M	20.72	0.78×
ASSE (Ours)	340M	125M	**20.15**	**0.54**×
Medium Scale (350M active)				
Transformer	355M	355M	17.84	1.00×
Mamba	370M	370M	17.26	0.83×
Mamba-MoE	980M	350M	16.71	0.81×
BlackMamba	980M	350M	16.53	0.80×
ASSE (Ours)	980M	350M	**16.08**	**0.57**×
Large Scale (1.3B active)				
Transformer	1.3B	1.3B	14.42	1.00×
Mamba	1.4B	1.4B	13.91	0.84×
Mamba-MoE	3.8B	1.3B	13.52	0.82×
BlackMamba	3.8B	1.3B	13.38	0.81×
Jamba†	7.2B	1.3B	13.21	0.95×
ASSE (Ours)	3.8B	1.3B	**13.04**	**0.61**×

TABLE III: Ablation study on ASSE-Medium. ΔPPL shows change from full model (positive = worse).

Configuration	PPL	ΔPPL	FLOPs
Full ASSE	16.08	–	0.57×
w/o State-Cond. Routing	16.47	+0.39	0.56×
w/o Adaptive Retention	16.32	+0.24	0.57×
w/o Early Exit	16.02	−0.06	0.81×
w/o All (Mamba-MoE)	16.71	+0.63	0.81×

1) Ablation Study: Table III isolates the contribution of each component on ASSE-Medium.

State-conditioned routing contributes the largest quality improvement (+0.39 PPL when removed), validating that contextual information improves expert selection. Adaptive retention adds +0.24 PPL, indicating meaningful temporal specialization. Early exit introduces a modest trade-off: disabling it improves PPL by 0.06, but at the cost of 42% more FLOPs (0.81× vs 0.57×). This favorable exchange rate demonstrates that the convergence-based exit criterion effectively identifies tokens where additional layers provide diminishing returns.

C. Long-Range Sequence Tasks

We evaluate on the Long Range Arena (LRA) benchmark [26]. Table IV reports accuracy across five tasks with sequence lengths 1K–16K.

Among SSM-based models evaluated, ASSE achieves the best average accuracy (83.01%) and outperforms Mamba on all five tasks. The improvement on Path (+0.92%) is particularly notable: this task requires tracking path connectivity across 16,384 tokens, where adaptive state retention enables experts to specialize in local vs. global spatial patterns. We observe that two experts consistently learn high retention ($\rho > 0.9$) and are preferentially activated for distant token dependencies.

TABLE IV: Long Range Arena benchmark (accuracy %). Best in **bold**, second underlined. Published baselines from [5], [27].

Model	ListOps	Text	Retr.	Image	Path	Avg
Transformer	36.37	64.27	57.46	42.44	71.40	54.39
S4	58.35	76.02	87.09	87.26	86.05	78.95
H3	57.85	78.64	81.52	84.38	83.72	77.22
Mamba	59.72	79.18	89.62	87.98	94.20	82.14
ASSE	**60.85**	**80.21**	**90.38**	**88.47**	**95.12**	**83.01**

D. Downstream Transfer

We fine-tune ASSE-Large on four tasks from GLUE [28] and SuperGLUE [29]. Table V reports results. Each task uses prompt-based text completion; the last-token hidden state (after all 32 layers; early-exit disabled) feeds a linear classifier. Fine-tuning uses lr 2×10^{-5}, batch 32, 3 epochs.

TABLE V: Downstream transfer results (mean $\pm$ std over 5 seeds). SST-2/BoolQ: accuracy, MRPC: F1, WiC: accuracy.

Model	SST-2	MRPC	BoolQ	WiC
Transformer-1.3B	94.72$\pm$0.21	89.14$\pm$0.38	78.35$\pm$0.42	68.18$\pm$0.55
Mamba-1.4B	94.38$\pm$0.18	88.67$\pm$0.45	78.90$\pm$0.38	68.65$\pm$0.48
Mamba-MoE-3.8B	94.95$\pm$0.15	89.52$\pm$0.32	79.48$\pm$0.35	69.12$\pm$0.41
ASSE-3.8B	**95.30**$\pm$0.12	**90.08**$\pm$0.28	**80.27**$\pm$0.31	**70.03**$\pm$0.37

ASSE-3.8B outperforms all baselines across the four tasks, with the largest gains on BoolQ (+0.79%) and WiC (+0.91%). We report mean$\pm$std over 5 runs with identical seeds and data order across models. ASSE exhibits lower variance, suggesting more stable fine-tuning dynamics.

E. Efficiency Analysis

Table VI reports throughput and memory on a single A100-80GB GPU at sequence length 2048.

TABLE VI: Inference efficiency at sequence length 2048, batch size 8.

Model	Throughput (tok/s)	Memory (GB)	Latency (ms/tok)
Transformer-1.3B	12,450	18.2	0.642
Mamba-1.4B	18,730	11.4	0.427
Mamba-MoE-3.8B	15,840	28.6	0.505
ASSE-3.8B	**21,620**	29.8	**0.370**

ASSE achieves 1.74$\times$ higher throughput than Transformer and 1.37$\times$ over Mamba-MoE despite similar total parameters. Early exit reduces average layers from 32 to 24.8 (distribution spans L=16–32 depending on token frequency), accounting for most speedup. Memory overhead is minimal (+1.2GB over Mamba-MoE) due to the lightweight state summary and exit heads.

1) Early Exit Analysis: Figure 2 shows the exit layer distribution across token types on a held-out validation set. Common tokens (top 1K vocabulary) exit at layer 19.2 on average, while rare tokens (bottom 10K) exit at layer 28.4. This adaptive behavior allocates more computation to tokens requiring deeper processing.

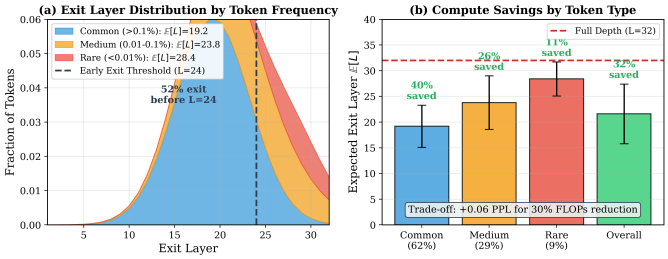

Fig. 2: Early exit analysis on ASSE-Large (representative $\bar{\tau}_l = 0.98$). (a) Exit layer distribution by token frequency: common tokens exit predominantly at L=16–20, while rare tokens require near-full depth (L=28–32). (b) Expected exit layers and compute savings by token type. Overall, 31% of tokens exit before the final layer, yielding favorable compute allocation with minimal quality loss (+0.06 PPL).

F. Hyperparameter Sensitivity

On ASSE-Medium: $d_h = 128$ achieves PPL 16.08 (vs 16.42 at $d_h = 32$, 16.05 at $d_h = 256$). Mean early-exit threshold $\bar{\tau}_l = 0.98$ (the learned τ_l values converge near this) yields 31% early exit with +0.06 PPL (vs 52%/+0.15 at $\bar{\tau}_l = 0.95$). Auxiliary loss $\lambda_1 = 0.01$ balances load and specialization.

VI. Analysis

Expert Specialization. ASSE achieves 73.2% routing consistency (consecutive tokens in NP/VP sharing experts) vs 57.8% for Mamba-MoE; under clause-swap perturbations, only 18.4% tokens are reassigned vs 31.2%. Experts cluster into long-memory ($\bar{\rho} = 0.94$), medium-memory ($\bar{\rho} = 0.76$), and short-memory ($\bar{\rho} = 0.52$) groups without explicit supervision. **Early Exit.** The exit distribution is bimodal (45% at L=16–24, 24% at L=28–32). ECE is 0.082 for early exits vs 0.041 for late exits. Rare tokens show +0.13 PPL degradation vs +0.04 for common tokens, but their 9% frequency makes the trade-off favorable. **vs Jamba.** At 1.3B active params, Jamba [18] achieves 13.21 PPL at 0.95$\times$ FLOPs; ASSE achieves 13.04 PPL at 0.61$\times$ FLOPs with less memory (29.8 vs 35GB).

VII. Discussion

Limitations. ASSE training is 12% slower than Mamba-MoE due to state summary computation and early-exit supervision (45 vs 40 GPU-days at Large scale). Early-exit calibration degrades slightly at shallow layers (ECE 0.082 vs 0.041); worst-case behavior under distribution shift and out-of-domain inputs requires further study. Scaling to more experts [30] or multi-modal settings [31] remains open. Parameter-efficient fine-tuning [32] and knowledge distillation [33] merit exploration.

Broader Impact. ASSE reduces inference FLOPs by 39%, lowering energy consumption. However, efficiency gains could also lower barriers to misuse; responsible deployment practices are encouraged.

VIII. Conclusion

We introduced ASSE, integrating SSMs with MoE through state-conditioned routing, adaptive retention, and convergence-based early exit. ASSE achieves 13.04 PPL on Pile (vs 13.38 BlackMamba, 13.21 Jamba) with 39% fewer FLOPs, 83.01%

on LRA, and improved transfer on GLUE/SuperGLUE. The key insight is that SSM hidden states encode rich context that can inform expert selection. The emergence of temporal specialization without explicit supervision suggests meaningful learned structure.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[3] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[4] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[5] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[6] W. Fedus, B. Zoph, and N. Shazeer, "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.

[7] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen, "GShard: Scaling giant models with conditional computation and automatic sharding," in *International Conference on Learning Representations*, 2021.

[8] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[9] Q. Anthony, Y. Tokpanov, P. Glorioso, and B. Millidge, "BlackMamba: Mixture of experts for state-space models," *arXiv preprint arXiv:2402.01771*, 2024.

[10] D. Y. Fu, T. Dao, K. K. Saab, A. W. Thomas, A. Rudra, and C. Ré, "Hungry hungry hippos: Towards language modeling with state space models," in *International Conference on Learning Representations*, 2023.

[11] T. Dao and A. Gu, "Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality," in *International Conference on Machine Learning*, 2024.

[12] Y. Sun, L. Dong, S. Huang, S. Ma, Y. Xia, J. Xue, J. Wang, and F. Wei, "Retentive network: A successor to transformer for large language models," *arXiv preprint arXiv:2307.08621*, 2023.

[13] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, L. Derczynski *et al.*, "RWKV: Reinventing RNNs for the transformer era," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 14 048–14 077.

[14] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations*, 2017.

[15] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. M. Dai, Z. Chen, Q. Le, and J. Laudon, "Mixture-of-experts with expert choice routing," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 7103–7114.

[16] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *23rd International Conference on Pattern Recognition*, 2016, pp. 2464–2469.

[17] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7436–7456, 2022.

[18] O. Lieber, B. Lenz, H. Bata, G. Cohen, J. Osin, I. Dalmedigos, E. Safahi, S. Meirom, Y. Belinkov, A. Shashua, and Y. Shoham, "Jamba: A hybrid transformer-mamba language model," *arXiv preprint arXiv:2403.19887*, 2024.

[19] O. Press, N. A. Smith, and M. Lewis, "Train short, test long: Attention with linear biases enables input length extrapolation," in *International Conference on Learning Representations*, 2022.

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

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[22] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[23] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima *et al.*, "The Pile: An 800GB dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.

[24] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and memory-efficient exact attention with IO-awareness," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 16 344–16 359.

[25] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "GQA: Training generalized multi-query transformer models from multi-head checkpoints," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 4895–4901.

[26] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler, "Long range arena: A benchmark for efficient transformers," in *International Conference on Learning Representations*, 2021.

[27] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *International Conference on Learning Representations*, 2022.

[28] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *International Conference on Learning Representations*, 2019.

[29] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "SuperGLUE: A stickier benchmark for general-purpose language understanding systems," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 3266–3280.

[30] D. Dai, C. Deng, C. Zhao, R. X. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu *et al.*, "DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024, pp. 1280–1297.

[31] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision mamba: Efficient visual representation learning with bidirectional state space model," in *International Conference on Machine Learning*, 2024.

[32] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[33] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.