

Dynamic Sparse Attention and Mixture-of-Experts for Iterative Visual Classification

Ahsan Umar

232552@icp.edu.pk

Islamia College University

Research Article

Keywords: Computer Vision, Conditional Computation, Dynamic Sparse Attention, Image Classification, Iterative Refinement, Mixture-of-Experts, Transformer

Posted Date: August 5th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10571274/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Dynamic Sparse Attention and Mixture-of-Experts for Iterative Visual Classification

Ahsan Umar 

Islamia College University, Peshawar, Pakistan
232552@icp.edu.pk

Abstract—Single-pass image classifiers cannot revise an uncertain prediction after forming it. This study investigates iterative visual classification with a compact transformer that repeatedly refines a class hypothesis using a global image embedding and an expanding history of latent states. Dynamic Sparse Attention (DSA) selects the top- k states once the history exceeds the attention budget, while a Mixture-of-Experts (MoE) layer activates two of four feed-forward experts per token. Standard Attention with a dense MLP, Standard Attention with MoE, and DSA with MoE are evaluated on CIFAR-10, CIFAR-100, MNIST, and FashionMNIST, with ablations over recursion depth, attention budget, routing, and prediction embedding. The selected DSA-MoE operating point achieves single-seed accuracies of 92.3%, 69.2%, 98.7%, and 92.3%, respectively. Relative to the dense reference, it reduces ideally active parameters from 13.13M to 7.10M and the representative transformer-block multiply-add ratio to $0.52\times$; measured training time decreases from 180 to 165 seconds per epoch. A moderate budget of $k = 6$ gives 36% average sparsity, whereas $k = 2$ lowers CIFAR-10 accuracy by 3.2 percentage points. These results indicate that recursive conditional computation can preserve competitive accuracy while reducing active computation, although multi-seed evaluation is required to establish the small comparative gains.

Index Terms—Computer Vision, Conditional Computation, Dynamic Sparse Attention, Image Classification, Iterative Refinement, Mixture-of-Experts, Transformer

I. INTRODUCTION

Most visual classifiers transform an image into a label in one pass. That design is efficient and effective, but it provides no opportunity to revisit an uncertain prediction. Iterative refinement offers a different view: a model can preserve its current hypothesis, compare it with the image representation and earlier latent states, and then update the prediction over several compact reasoning steps. This idea has proved useful in recurrent networks, structured prediction, and recent recursive reasoning models [6, 7, 20].

Transferring recursive refinement from symbolic grids to natural images raises two questions. First, the model needs a visual representation that is compact enough to revisit repeatedly. Second, repeated attention over an expanding history increases computation unless the model can select which states and parameters are relevant at each step. We address the first question with a lightweight convolutional encoder and the second with two complementary forms of conditional computation: Dynamic Sparse Attention (DSA), which selects a subset of tokens, and Mixture-of-Experts (MoE) routing, which selects a subset of feed-forward parameters.

The resulting model maintains three kinds of information: a global image embedding x_{feat} , a class-prediction token y_t , and a latent history $Z_t = (z_0, \dots, z_t)$. The history grows with recursion depth, so the attention sequence has length $L_t = t + 2$. This detail is central to the design. A top- k mechanism is dense whenever $L_t \leq k$ and becomes genuinely sparse only after the history exceeds the attention budget. The model therefore moves naturally from dense early refinement to increasingly selective later refinement.

We study this architecture on CIFAR-10, CIFAR-100, MNIST, and FashionMNIST. Rather than treating accuracy,

parameter count, and computation as interchangeable measures, we distinguish stored parameters, parameters active per token, theoretical multiply-add counts, and measured training time. The experiments are complemented by ablations over recursion depth, attention budget, expert routing, and prediction embedding.

A. Research Questions

The study is organized around four questions:

- 1) Can an $x/y/z$ recursive formulation support iterative visual classification across datasets of different difficulty?
- 2) At what recursion steps does top- k attention become sparse, and how does the budget k affect accuracy and computation?
- 3) How do expert routing and attention selection change stored capacity, active computation, and measured training time?
- 4) What aspects of the internal trajectory are observable when the visual input is represented by one global token rather than spatial patches?

B. Contributions

The main contributions are:

- 1) a dimensionally explicit recursive vision model that connects a global image representation, a prediction token, and a growing latent history;
- 2) a DSA formulation whose sparsity changes with recursion depth, together with an analytical description of the exact dense-to-sparse crossover;
- 3) a unified comparison of dense attention, expert routing, and their combination, supported by ablations over the principal architectural choices; and

- 4) a careful interpretation of internal token and routing trajectories that distinguishes observable refinement dynamics from unsupported spatial explanations.

The central claim is deliberately focused: recursive conditional computation can preserve competitive classification accuracy while reducing active computation, but the magnitude and statistical reliability of the gains depend on recursion depth, sparsity budget, and routing configuration.

II. RELATED WORK

This work sits at the intersection of four research threads: iterative and recurrent reasoning, sparse attention mechanisms, conditional computation through Mixture-of-Experts routing, and efficient convolutional and transformer vision backbones. We review each thread in turn, then explain precisely how our design combines and departs from them.

A. Iterative and Recurrent Reasoning

The idea that a model should revise its own output rather than commit to a single forward pass has a long history in neural sequence modeling. Elman networks demonstrated that recurrent updates to a hidden state could capture temporal structure that a strictly feed-forward network cannot [19]. Long Short-Term Memory (LSTM) networks introduced gating mechanisms that made such recurrence trainable over much longer sequences by mitigating vanishing gradients [21]. The Transformer architecture replaced recurrence with parallelizable self-attention [9], but subsequent work has reintroduced controlled looping on top of transformer blocks to recover the benefits of iterative refinement without abandoning parallel training.

In computer vision specifically, iterative refinement has appeared in pose estimation and structured prediction, where a model repeatedly corrects an initial estimate using both the input and its own prior output; such staged refinement has been shown to reduce localization error substantially relative to a single-shot regression [20]. In natural language processing, the same principle underlies Chain-of-Thought prompting, which asks a model to produce visible intermediate reasoning steps [3], zero-shot variants that elicit similar behavior without task-specific exemplars [5], and test-time compute scaling, which trades additional inference-time search for accuracy [4]. These methods are effective but remain constrained by the sequential, token-by-token nature of autoregressive decoding: even with substantial additional inference-time computation, frontier reasoning models continue to score in the single digits to low tens of percent on the hardest tier of the ARC-AGI abstraction benchmark, with the exact figure depending sensitively on model version, reasoning budget, and evaluation date [1, 2].

HRM addresses this differently, using a dual-timescale recurrent transformer rather than autoregressive decoding [6]. Its low-level module f_L updates on every inner step while its high-level module f_H updates once every n inner steps, across T outer supervisory phases; in the reported configuration $n = 2$ and $T = 2$. Follow-up analysis attributes a large share of HRM’s benchmark gains to deep supervision, a training

scheme that attaches a loss to every intermediate reasoning state rather than only the final one, which alone raised ARC-AGI accuracy from 19% to 39% [8]. TRM [7] simplifies this further into a single small transformer with an explicit $x/y/z$ recurrence: an encoded input x , an output hypothesis y , and a latent reasoning token z that is refined for N_{step} iterations via $z \leftarrow \text{Transformer}(x, y, z)$ followed by $y \leftarrow \text{Proj}(z)$. TRM’s own ablations show that both deep supervision and recurrent (rather than single-step) refinement are necessary to reach its reported 45% ARC-AGI-1 and 8% ARC-AGI-2 accuracy.

Our architecture inherits this $x/y/z$ recurrence directly from TRM but changes what each symbol represents: x_{feat} is a convolutional image embedding rather than a discrete grid encoding, y is a class-prediction token rather than a cell-level output, and z is a latent visual-reasoning state whose history accumulates across iterations rather than being overwritten at each step. Unlike HRM’s two-module, two-timescale design, our model uses a single Dynamic Sparse Attention mechanism operating over an explicitly growing sequence, which we argue offers a more transparent trace of intermediate reasoning than the token-by-token generation used in autoregressive LLM reasoning [22].

B. Sparse Attention Mechanisms

Standard self-attention computes an $O(L^2d)$ interaction over a sequence of length L and embedding width d , which becomes the dominant cost as L grows. A substantial body of work reduces this cost through structured or learned sparsity. BigBird combines random, local (sliding-window), and a small number of global tokens to achieve near-linear $O(L)$ complexity while retaining a theoretical universal-approximation guarantee [23]. Longformer similarly combines a local attention window with a small set of task-informed global tokens [22]. The Sparse Transformer factorizes attention into strided and local patterns to scale to very long sequences [24]. These methods share a common trait: the sparsity pattern is fixed by design (which positions attend to which) rather than learned from the input.

Content-dependent sparsity, where the model itself decides which tokens deserve attention, has recently been popularized at scale by DeepSeek’s sparse-attention line of work. DeepSeek-V2 introduced Multi-head Latent Attention (MLA), which compresses the key-value cache through a low-rank latent projection to reduce memory bandwidth [10], but MLA does not by itself skip attending to any token; every query still attends to the full compressed key-value set. The specific mechanism that motivates this study, a lightweight indexer that scores candidate keys and restricts full attention computation to the top- k entries, was introduced later in DeepSeek-V3.2 as DeepSeek Sparse Attention [11]. We adopt the same two-stage score-then-select principle for a short, growing sequence of visual reasoning tokens. Because our indexer assigns one query-independent relevance score per token, it is a simplified small-sequence analogue rather than an exact counterpart of the per-query formulation.

The efficiency benefit of any top- k attention mechanism is bounded by how much larger L is than k . When L is

small, as in the first iteration of the proposed recurrence, top- k selection cannot prune anything because the budget is clipped to the sequence length. Section III therefore derives the full sequence-length trajectory and identifies the exact iterations at which genuine sparsification begins.

C. Mixture-of-Experts and Conditional Computation

Mixture-of-Experts (MoE) layers replace a single dense transformation with a bank of E smaller expert sub-networks and a learned gate that routes each token to a small subset of them, typically the top $m \ll E$ experts by gate score [15, 27]. This decouples a model’s total parameter count from its per-token computational cost: total capacity scales with E , while active compute scales with m . GShard demonstrated that this style of routing could scale to models with hundreds of billions of parameters using automatic sharding across accelerators [28], and the Switch Transformer showed that routing to a single expert ($m = 1$) per token, rather than the more common $m = 2$, could simplify the routing computation with a comparatively small accuracy cost while improving pretraining speed [29].

A well-known failure mode of MoE training is routing collapse, where the gate learns to send most tokens to a small subset of experts, wasting the capacity of the remainder; auxiliary load-balancing losses that encourage uniform expert utilization are now standard practice to mitigate this [15, 29]. Our Mixture-of-Experts layer follows this convention directly: it uses $E = 4$ experts with top- $m = 2$ routing and a load-balancing penalty, replacing the model’s feed-forward sub-layer rather than its attention sub-layer. Because our experts are also narrower (hidden width d) than the dense feed-forward layer they replace (hidden width $4d$), our reported efficiency gains reflect two conflated effects, a per-expert width reduction and a routing-fraction reduction, that must be reported separately; we return to this point with an explicit derivation in Section IV.

D. Efficient Vision Architectures

Efficient convolutional design has a long history of trading a small amount of accuracy for substantial reductions in parameters and compute. MobileNet introduced depthwise-separable convolutions to approximate standard convolutions at a fraction of the multiply-add cost [17], and MobileNetV2 extended this with inverted residual blocks and linear bottlenecks [30]. EfficientNet proposed a principled compound scaling rule that jointly scales network depth, width, and input resolution, showing that naively scaling only one dimension is suboptimal [18]. Vision Transformers (ViTs) took a different route, applying self-attention directly to flattened image patches and matching or exceeding convolutional baselines at sufficient pretraining scale [16]; the Swin Transformer subsequently reintroduced locality and hierarchy through shifted local attention windows to improve efficiency on dense prediction tasks [31]. Detection-oriented transformer variants such as DETR [32] and Deformable DETR [33] demonstrate that attention-based reasoning over a moderate number of visual tokens can replace hand-designed region-proposal pipelines,

which is architecturally closer to our own setting of attention over a small token set.

Our convolutional front end is deliberately conventional: three convolution–ReLU–pooling stages followed by global average pooling, in the spirit of the compact backbones above, chosen so that the recursive stage, not the feature extractor, is the object of study. Unlike ViT-style patch tokenization, which produces tens to hundreds of visual tokens and is the setting in which DSA-style sparsity provides the largest relative savings, our backbone collapses each image to a single embedding vector x_{feat} . This choice keeps the token-role recurrence (x_{feat} , y , and the growing z history) small and interpretable at the level of token roles, but it also means, as discussed in Section VI, that the attention traces cannot be mapped back to spatial image regions.

E. Summary and Positioning

In summary, our design combines the TRM-style $x/y/z$ recurrence [7], an indexer-based top- k attention mechanism inspired by DeepSeek Sparse Attention [11], and a top- m -of- E Mixture-of-Experts feed-forward layer [15, 29] on a compact convolutional backbone [17, 18]. The distinctive element is their combination within a visual recurrence whose sequence length grows over time, making the relationship between recursion depth and conditional computation explicit.

III. METHODOLOGY

This section specifies the proposed architecture and introduces each symbol with its dimensionality. Figure 1 gives the end-to-end data flow. We first define the shared recurrence (Section III-A), then the convolutional feature extractor (Section III-B), Dynamic Sparse Attention (Section III-C), the Mixture-of-Experts layer (Section III-D), and the three evaluated component configurations (Section III-E).

A. The Shared $x/y/z$ Recurrence

All three model variants share the same outer recurrence and differ only in the attention and feed-forward sub-layers inside each transformer block.

Step 0: feature extraction and initial hypothesis. A convolutional backbone f_{CNN} (detailed in Section III-B) maps the input image to a single global embedding,

$$x_{\text{feat}} = f_{\text{CNN}}(x) \in \mathbb{R}^d. \quad (1)$$

A linear layer produces initial class logits $\ell_0 = W_{\text{direct}}x_{\text{feat}} + b_{\text{direct}} \in \mathbb{R}^C$, where $W_{\text{direct}} \in \mathbb{R}^{C \times d}$. Depending on the variant, the initial prediction token $y_0 \in \mathbb{R}^d$ is obtained either as a *soft embedding*, a probability-weighted mixture of learned class embedding vectors $E_c \in \mathbb{R}^d$,

$$y_0 = \sum_{c=1}^C \text{softmax}_c(\ell_0) E_c, \quad (2)$$

or as a *hard embedding*, which looks up the single embedding of the arg-max class,

$$y_0 = E_{\arg \max(\ell_0)}. \quad (3)$$

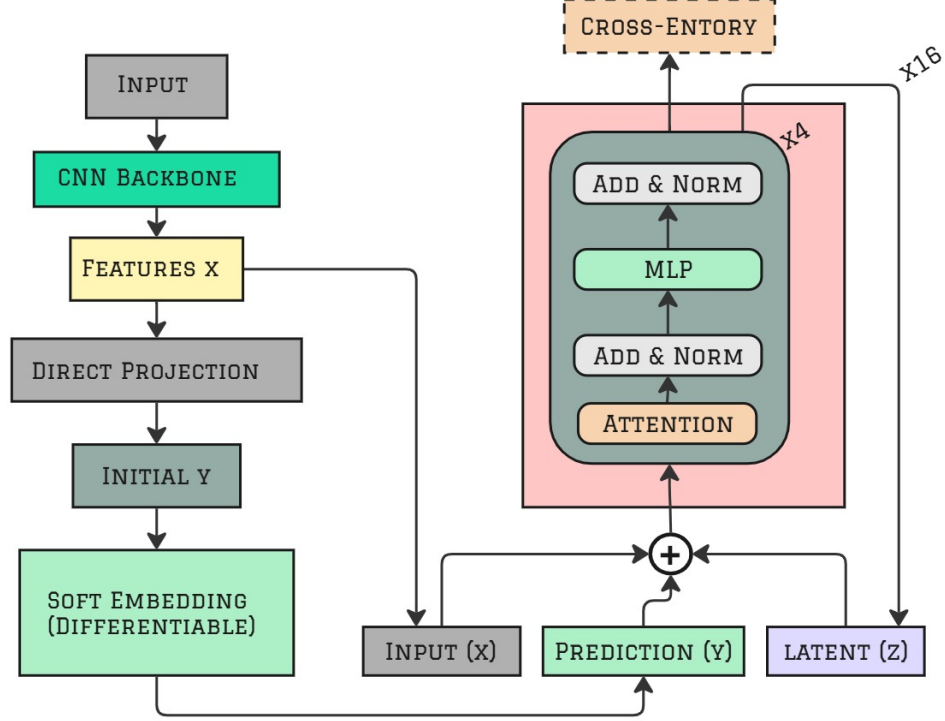


Fig. 1. Conceptual overview of the shared architecture: a CNN produces the visual representation and initial prediction token, after which a four-block transformer iteratively refines the prediction and latent state. The diagram shows the baseline attention and MLP sub-layers; DSA replaces attention and MoE replaces the MLP in the corresponding variants. It is schematic: the evaluated model uses the concatenated growing sequence $[x_{\text{feat}}; y_{t-1}; Z_{t-1}]$ defined in Eq. (5), rather than an elementwise sum, and the equations below specify the exact update order.

The hard variant is not differentiable through the $\arg \max$; we use a straight-through estimator (STE) [36], which copies the gradient of y_0 with respect to the downstream loss directly back onto ℓ_0 during the backward pass, exactly as if Eq. (2) had been used, while the forward pass still uses the discrete lookup of Eq. (3).

Step 1: latent-history initialization. The latent buffer is seeded with a single entry equal to the initial prediction token, $Z_0 = (z_0)$ with $z_0 := y_0$. This gives the recurrence an input-dependent initial state rather than a fixed zero vector.

Step $t = 1, \dots, N_{\text{step}}$: growing-sequence refinement. At every recursion step t , the transformer input is the concatenation of the image embedding, the current prediction token, and the *entire accumulated latent history so far*:

$$h_t = [x_{\text{feat}}; y_{t-1}; z_0, z_1, \dots, z_{t-1}] \in \mathbb{R}^{L_t \times d}, \quad (4)$$

augmented with a sinusoidal positional encoding [9] indexed by token position within h_t . Because Z_0 contributes one entry and one new latent token is appended to the history at the end of every subsequent step, the sequence length follows

$$L_t = 2 + t, \quad t = 1, \dots, N_{\text{step}}, \quad (5)$$

i.e., $L_1 = 3$ at the first step and $L_{16} = 18$ at the sixteenth. Retaining the full history is what allows a top- k budget to become selective: the operation is equivalent to dense attention whenever $L_t \leq k$ and becomes genuinely sparse only when $L_t > k$. Section IV gives the crossover point for each evaluated budget.

The sequence h_t is passed through $B = 4$ stacked transformer blocks (Section III-E specifies each block’s internal attention and feed-forward mechanism); we write the composition of all B blocks as $T(\cdot)$. The block stack outputs a refined sequence of the same length L_t , from which we take the final position as the newly refined latent token:

$$z_t := \text{last}(T(h_t)) \in \mathbb{R}^d, \quad (6)$$

$$Z_t := Z_{t-1} \parallel (z_t). \quad (7)$$

The class-prediction token is then updated by a linear read-out (“reverse embedding”) from the new latent token,

$$\ell_t = W_{\text{out}} z_t + b_{\text{out}} \in \mathbb{R}^C, \quad (8)$$

$$y_t = \sum_{c=1}^C \text{softmax}_c(\ell_t) E_c \quad (\text{or the hard-embedding form}). \quad (9)$$

After N_{step} iterations, the final logits $\ell_{N_{\text{step}}}$ are used both for the loss during training and for the predicted class at inference. The full recurrence is summarized in Algorithm 1.

Complexity and role of Algorithm 1. This outer loop runs in $O(N_{\text{step}})$ sequential iterations, each dominated by the cost of the B transformer blocks operating on a sequence of length L_t ; because L_t grows linearly with t , the total cost across all N_{step} iterations is $O\left(B \sum_{t=1}^{N_{\text{step}}} c(L_t)\right)$, where $c(L)$ is the per-block cost at sequence length L (specified separately for each attention/FFN combination in Section IV). This recurrence is

Algorithm 1 Iterative Visual Refinement (shared recurrence)

Require: image x ; recursion depth N_{step} ; embedding table $E \in \mathbb{R}^{C \times d}$; embedding mode $\in \{\text{soft}, \text{hard}\}$

Ensure: final logits $\ell_{N_{\text{step}}}$

```

1:  $x_{\text{feat}} \leftarrow f_{\text{CNN}}(x)$ 
2:  $\ell_0 \leftarrow W_{\text{direct}} x_{\text{feat}} + b_{\text{direct}}$ 
3:  $y_0 \leftarrow \text{EMBED}(\ell_0, \text{mode})$  {Eq. (2) or (3); STE if hard}
4:  $Z \leftarrow [y_0]$  {latent history buffer, seeded with  $z_0 := y_0$ }
5: for  $t = 1$  to  $N_{\text{step}}$  do
6:    $h_t \leftarrow \text{CONCAT}(x_{\text{feat}}, y_{t-1}, Z)$  {length  $L_t = 2 + t$ , Eq. (5)}
7:    $h_t \leftarrow h_t + \text{POSITIONALENCODING}(h_t)$ 
8:   for  $b = 1$  to  $B$  do
9:      $h_t \leftarrow \text{TRANSFORMERBLOCK}_b(h_t)$  {attention + FFN sub-layer, Sec. III-E}
10:  end for
11:   $z_t \leftarrow h_t[\text{last position}]$ 
12:   $Z \leftarrow \text{APPEND}(Z, z_t)$ 
13:   $\ell_t \leftarrow W_{\text{out}} z_t + b_{\text{out}}$ 
14:   $y_t \leftarrow \text{EMBED}(\ell_t, \text{mode})$ 
15: end for
16: return  $\ell_{N_{\text{step}}}$ 

```

the single point of comparison across all three model variants; every other design choice discussed below is a substitution for the internals of TRANSFORMERBLOCK in line 9, leaving the outer loop, the embedding rule, and the recursion depth identical across variants unless explicitly varied in an ablation.

Training objective. All variants are trained with label-smoothed cross-entropy on the final logits,

$$\mathcal{L} = - \sum_{c=1}^C \tilde{y}_c \log(\text{softmax}_c(\ell_{N_{\text{step}}})) \quad (10)$$

$$\tilde{y}_c = (1 - \epsilon) \delta_{c=y^*} + \frac{\epsilon}{C} \quad (11)$$

with smoothing factor $\epsilon = 0.05$, following standard practice for stabilizing classifier training [35]. Supervision is applied only to the terminal logits $\ell_{N_{\text{step}}}$; intermediate-step supervision is considered separately in Section VII. For MoE configurations, the total objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L} + \lambda_{\text{load}} \mathcal{L}_{\text{load}}, \quad (12)$$

where $\lambda_{\text{load}} = 0.01$ and $\mathcal{L}_{\text{load}}$ penalizes imbalanced expert utilization.

B. Convolutional Feature Extractor

The feature extractor f_{CNN} is identical across all three variants and follows a conventional compact design [17, 18]. It contains three stages, each comprising a 3×3 convolution (padding 1), ReLU activation, and 2×2 max pooling with stride 2, with channel widths $3 \rightarrow 64 \rightarrow 128 \rightarrow 256$. Global adaptive average pooling then reduces the final feature map to a 256-dimensional vector, and a linear projection maps it to the embedding width $d = 512$. This pooling choice limits the parameter count and gives a translation-tolerant global representation [34], but it also removes spatial coordinates before the recursive stage.

C. Dynamic Sparse Attention (DSA)

DSA replaces the query-key-value attention core of a standard transformer block with a variant that restricts each layer's attention computation to a learned subset of the available tokens, in the spirit of the lightning-indexer top- k mechanism used in DeepSeek Sparse Attention [11], adapted here to a short, growing sequence rather than a long language-model context.

Given an input sequence $h \in \mathbb{R}^{L \times d}$ (we drop the step subscript t in this subsection for readability), DSA computes per-head queries in the usual way, $Q = hW_Q \in \mathbb{R}^{L \times d}$, reshaped to H heads of width $d_k = d/H$. Unlike standard multi-head attention, the key and value projections are *shared across heads* rather than computed per head, $K = hW_K \in \mathbb{R}^{L \times d_k}$ and $V = hW_V \in \mathbb{R}^{L \times d_k}$, which is a multi-query-attention-style simplification [26] that reduces the projection cost at a small representational cost, since every head attends to the same key/value space.

A lightweight two-layer indexer network $g_{\text{idx}} : \mathbb{R}^{d_k} \rightarrow \mathbb{R}$ with hidden width $d_k/4$ scores each token's relevance from a head-averaged query representation, $\bar{Q} = \text{mean}_h(Q) \in \mathbb{R}^{L \times d_k}$:

$$s_j = g_{\text{idx}}(\bar{Q}_j), \quad j = 1, \dots, L. \quad (13)$$

Equation (13) produces one relevance score per token, shared across all query positions, rather than a query-specific score s_{ij} . This query-independent indexer is appropriate to the short sequences studied here and is simpler than the per-query formulation used in DeepSeek Sparse Attention.

The top- k operation then selects the k tokens with the largest scores, using an explicit descending sort:

$$\mathcal{I} = \underset{\text{desc}}{\text{argsort}}(s)[1 : k'], \quad k' = \min(k, L). \quad (14)$$

The clipping $k' = \min(k, L)$ guarantees that the operation is well-defined. Whenever the configured budget is not smaller than the current sequence length L_t (Eq. (5)), $\mathcal{I}$ contains every token and no pruning occurs. Every query position uses the same selected index set $\mathcal{I}$, gathering the corresponding rows of K and V ,

$$K_{\mathcal{I}} = K[\mathcal{I}, :] \in \mathbb{R}^{k' \times d_k}, \quad V_{\mathcal{I}} = V[\mathcal{I}, :] \in \mathbb{R}^{k' \times d_k}, \quad (15)$$

and computing standard scaled dot-product attention restricted to these rows, independently per head $r = 1, \dots, H$:

$$\text{DSA-Attn}_r(h) = \text{softmax}\left(\frac{Q_r K_{\mathcal{I}}^T}{\sqrt{d_k}}\right) V_{\mathcal{I}} \in \mathbb{R}^{L \times d_k}. \quad (16)$$

The H per-head outputs are concatenated and passed through an output projection $W_O \in \mathbb{R}^{d \times d}$ to produce the sub-layer's output. Algorithm 2 summarizes the full DSA forward pass.

Complexity. Standard multi-head attention requires four $d \times d$ projections, giving $O(4Ld^2)$ projection cost, and $O(2HL^2d_k)$ for score and value aggregation. DSA retains $d \times d$ query and output projections but shares $d \times d_k$ key and value projections across heads, giving $O(2Ld^2 + 2Ldd_k)$ projection cost. Its restricted attention core costs $O(2HLk'd_k)$, where $k' = \min(k, L)$. The gain therefore comes from both shared key/value projections and token selection; at the short

Algorithm 2 Dynamic Sparse Attention (DSA), one layer

Require: sequence $h \in \mathbb{R}^{L \times d}$; head count H ; budget k
Ensure: attention output $o \in \mathbb{R}^{L \times d}$

- 1: $Q \leftarrow hW_Q$; reshape to H heads of width $d_k = d/H$
- 2: $K \leftarrow hW_K \in \mathbb{R}^{L \times d_k}$; $V \leftarrow hW_V \in \mathbb{R}^{L \times d_k}$ {shared across heads}
- 3: $\bar{Q} \leftarrow \text{mean}_{\text{heads}}(Q) \in \mathbb{R}^{L \times d_k}$
- 4: $s \leftarrow g_{\text{idx}}(\bar{Q}) \in \mathbb{R}^L$ {Eq. (13)}
- 5: $k' \leftarrow \min(k, L)$
- 6: $\mathcal{I} \leftarrow \text{argsort}_{\text{desc}}(s)[1:k']$ {Eq. (14)}
- 7: $K_{\mathcal{I}} \leftarrow K[\mathcal{I}, :]$; $V_{\mathcal{I}} \leftarrow V[\mathcal{I}, :]$
- 8: **for** $r = 1$ **to** H **do**
- 9: $o_r \leftarrow \text{softmax}(Q_r K_{\mathcal{I}}^\top / \sqrt{d_k}) V_{\mathcal{I}}$
- 10: **end for**
- 11: $o \leftarrow \text{CONCAT}(o_1, \dots, o_H) W_O$
- 12: **return** o

sequence lengths studied here, projection and feed-forward operations remain the dominant terms.

D. Mixture-of-Experts (MoE) Feed-Forward Layer

The MoE layer replaces a transformer block’s dense feed-forward sub-layer with E smaller expert networks and a learned gate, following the standard sparsely-gated formulation [15]. For an input token $u \in \mathbb{R}^d$ (a single row of h), a gate produces routing logits $g = W_{\text{gate}} u \in \mathbb{R}^E$, and the top- m experts by gate score are activated:

$$p_e = \text{softmax}_e(g), \quad (17)$$

$$\mathcal{M} = \text{argsort}_{\text{desc}}(p)[1:m], \quad (18)$$

$$\tilde{g}_e = \frac{p_e}{\sum_{e' \in \mathcal{M}} p_{e'}} \quad \forall e \in \mathcal{M}. \quad (19)$$

where $\tilde{g}$ renormalizes the selected gate weights to sum to one. Each expert e is a two-layer network with hidden width $d_{\text{ff}}^{\text{exp}}$,

$$\text{Expert}_e(u) = W_2^{(e)} \text{GELU}(W_1^{(e)} u), \quad (20)$$

$$W_1^{(e)} \in \mathbb{R}^{d_{\text{ff}}^{\text{exp}} \times d}, \quad W_2^{(e)} \in \mathbb{R}^{d \times d_{\text{ff}}^{\text{exp}}}. \quad (21)$$

and the layer’s output is the gate-weighted sum over only the selected experts,

$$\text{MoE}(u) = \sum_{e \in \mathcal{M}} \tilde{g}_e \text{Expert}_e(u). \quad (22)$$

An auxiliary load-balancing loss, following [15, 29], penalizes uneven long-run expert utilization across a training batch, discouraging the gate from routing most tokens to a small subset of experts (a well-documented MoE failure mode known as routing collapse). We report its coefficient λ_{load} and its effect on utilization balance in Section V. Algorithm 3 summarizes the layer.

Active computation. A two-layer dense feed-forward network of hidden width d_{ff} costs approximately $2d d_{\text{ff}}$ multiply-

Algorithm 3 Mixture-of-Experts (MoE) feed-forward layer, one token

Require: token $u \in \mathbb{R}^d$; experts $\{\text{Expert}_e : 1 \leq e \leq E\}$; active count m ; gate W_{gate}
Ensure: output $o \in \mathbb{R}^d$

- 1: $g \leftarrow W_{\text{gate}} u \in \mathbb{R}^E$
- 2: $p \leftarrow \text{softmax}(g)$
- 3: $\mathcal{M} \leftarrow \text{argsort}_{\text{desc}}(p)[1:m]$
- 4: $\tilde{g}_e \leftarrow p_e / \sum_{e' \in \mathcal{M}} p_{e'}$ for $e \in \mathcal{M}$
- 5: $o \leftarrow \mathbf{0}$
- 6: **for** $e \in \mathcal{M}$ **do**
- 7: $o \leftarrow o + \tilde{g}_e \text{Expert}_e(u)$
- 8: **end for**
- 9: **return** o

adds per token. A routed layer evaluates m two-layer experts of width $d_{\text{ff}}^{\text{exp}}$, so the active-cost ratio is

$$\frac{\text{active MoE cost}}{\text{dense cost}} = \frac{m}{E} \frac{E d_{\text{ff}}^{\text{exp}}}{d_{\text{ff}}} \quad (23)$$

$$= \frac{m d_{\text{ff}}^{\text{exp}}}{d_{\text{ff}}}. \quad (24)$$

For $m = 2$, $d_{\text{ff}}^{\text{exp}} = d$, and $d_{\text{ff}} = 4d$, the active expert computation is one half of the dense feed-forward computation. The routing fraction m/E describes the fraction of stored experts used, whereas Eq. (24) also accounts for expert width; the two quantities coincide numerically here because $E d_{\text{ff}}^{\text{exp}} = d_{\text{ff}}$.

E. The Three Evaluated Variants

The study considers three component configurations. They share the convolutional backbone, embedding width $d = 512$, four transformer blocks, and eight attention heads. Each block uses residual connections and two LayerNorm operations. Recursion depth and prediction embedding are stated explicitly for each experiment.

Standard Attention + MLP (SA-MLP). Each transformer block uses dense multi-head self-attention with $H = 8$ heads of width $d_k = 64$, followed by a two-layer feed-forward network of hidden width $4d$. This is the dense reference configuration.

Standard Attention + MoE (SA-MoE). The attention sub-layer remains dense, while the feed-forward sub-layer is replaced by $E = 4$ experts of hidden width d with top- $m = 2$ routing. Matched comparisons with SA-MLP isolate the effect of expert routing.

Dynamic Sparse Attention + MoE (DSA-MoE). The feed-forward layer is identical to SA-MoE, while attention uses the DSA mechanism of Section III-C. Matched comparisons with SA-MoE isolate the attention mechanism, and the budget ablation in Section V-D1 shows how pruning strength changes the trade-off.

The cross-dataset table reports the selected operating point for each configuration and therefore summarizes attainable performance rather than a causal comparison. The subsequent ablations hold the relevant settings fixed when studying recursion depth, sparsity budget, expert routing, and embedding strategy.

TABLE I

PARAMETER COMPOSITION AND FULL-MODEL TOTALS FOR $d = 512$, $H = 8$, $B = 4$, AND $C = 10$. THE SHARED CNN AND PREDICTION HEADS CONTAIN 502,400 AND 15,380 PARAMETERS, RESPECTIVELY; EACH BLOCK ALSO CONTAINS 2,048 NORMALIZATION PARAMETERS. ACTIVE TOTALS ASSUME THAT UNROUTED EXPERTS ARE SKIPPED.

Model Variant	Attention/Block	Feed-Forward/Block	Stored Total	Active Total
SA-MLP	1,050,624	2,099,712	13,127,316	13,127,316
SA-MoE	1,050,624	2,103,300	13,141,668	8,939,172
DSA-MoE	592,033	2,103,300	11,307,304	7,104,808

IV. COMPUTATIONAL CHARACTERISTICS

We distinguish *stored parameters*, which determine model memory, from *active parameters*, which are applied to a token during a forward pass. These quantities are identical for a dense layer but differ under expert routing. We also separate projection cost, attention-core cost, and feed-forward cost because DSA and MoE affect different components.

A. Parameter Counts by Configuration

The SA-MLP and SA-MoE configurations have nearly equal stored capacity because four width- d experts collectively contain approximately the same feed-forward parameters as one width- $4d$ dense MLP. DSA-MoE stores fewer parameters because its shared key and value projections are narrower than the corresponding standard-attention projections.

B. Active Parameters and the MoE Ratio

Because only m of E experts are evaluated, a token activates $m \times 525,312$ expert parameters plus the 2,052-parameter gate. For $E = 4$ and $m = 2$, the active MoE feed-forward count is therefore 1,052,676 parameters per block, compared with 2,103,300 stored. Including the shared backbone, heads, attention, and normalization gives the full-model active counts in Table I. Stored parameters determine memory, whereas active parameters describe the idealized routed computation.

C. Sequence Length and the DSA Sparsity Crossover

Because the input sequence grows as $L_t = t + 2$ (Eq. (5)), DSA is dense whenever $L_t \leq k$ and first becomes sparse at step $t = k - 1$. Consequently, budgets $k = 4, 6$, and 8 first prune tokens at steps 3, 5, and 7, respectively.

For a 16-step recursion, the $k = 6$ configuration is dense for the first four iterations and sparse for the remaining twelve. Its per-step sparsity grows from $1 - 6/7 \approx 14\%$ at step 5 to $1 - 6/18 \approx 67\%$ at step 16. Efficiency must therefore be interpreted as a recursion-averaged effect rather than a fixed layer property.

D. Per-Component FLOP Estimates

Table II summarizes the leading per-block multiply-add costs. The initial classifier is included separately because it maps from d dimensions to C classes and therefore costs $O(dC)$.

At $L = 10$, $d = 512$, $H = 8$, and $k = 6$, the estimated per-block totals are approximately 31.56 million multiply-adds for SA-MLP, 21.09 million for SA-MoE, and 16.48 million for

DSA-MoE. Relative to SA-MLP, these correspond to $1.00\times$, $0.67\times$, and $0.52\times$. The DSA reduction includes narrower shared key/value projections as well as token pruning, while the MoE reduction reflects activation of two width- d experts instead of one width- $4d$ dense feed-forward layer. These theoretical ratios describe the transformer block and need not translate proportionally to end-to-end wall-clock time.

V. EXPERIMENTS

A. Datasets and Evaluation Protocol

We evaluate on four standard image classification benchmarks: CIFAR-10 and CIFAR-100 [14], MNIST [12], and FashionMNIST [13].

Splits. Each official training partition supplies the training and validation data: 50,000 images for CIFAR-10/100 and 60,000 for MNIST/FashionMNIST. The final 20% of that partition is held out for validation, leaving 80% for parameter optimization. The official 10,000-image test partition of each dataset is used only for final evaluation. Model selection is based exclusively on validation loss.

Preprocessing. CIFAR-10/100 images are normalized with ImageNet-style statistics (mean $[0.485, 0.456, 0.406]$, std $[0.229, 0.224, 0.225]$) and augmented at training time with random horizontal flips and random 32×32 crops from a 4-pixel-padded image. MNIST and FashionMNIST are converted to 3-channel (replicated grayscale) and normalized with mean/std of 0.1307/0.3081; training-time augmentation is limited to a small random rotation.

Optimization. All variants are trained with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-4}), an initial learning rate of 10^{-3} , and a validation-loss plateau schedule with patience 3 and reduction factor 0.5. The batch size is 128. Training runs for at most 100 epochs with early-stopping patience 5 and minimum validation-loss improvement $\delta = 0.005$. Gradient norms are clipped at 1.0, and label smoothing uses $\epsilon = 0.05$.

Model selection. The model state with the lowest validation loss is selected, and its accuracy on the held-out test partition is reported in Table III. Any validation accuracy discussed in this section refers to that same selected state.

Hardware and randomization. Experiments use one RTX 3090 GPU and random seed 42 with deterministic operations where available. Every accuracy value is therefore a single-seed measurement. Differences of 0.1–1.2 percentage points should be interpreted as directional rather than statistically established.

Architectural settings. All configurations use embedding width $d = 512$, $B = 4$ transformer blocks, and $H = 8$ attention heads. The recursion-depth ablation evaluates

TABLE II
LEADING MULTIPLY-ADD COSTS AT SEQUENCE LENGTH L , WITH $d_k = d/H$, $k' = \min(k, L)$, DENSE WIDTH $d_{\text{ff}} = 4d$, AND EXPERT WIDTH $d_{\text{ff}}^{\text{exp}}$.

Component	Standard Attention	Dynamic Sparse Attention
Initial classification projection		$O(dC)$
Attention projections (Q,K,V,O)	$O(4Ld^2)$	$O(2Ld^2 + 2Ldd_k)$
Attention core (scores + weighted values)	$O(2HL^2d_k)$	$O(2HLk'd_k)$
Indexer scoring	—	$O(Ld_k^2/4)$ (small relative to projections)
Feed-forward, dense ($d_{\text{ff}} = 4d$)		$O(2Ld d_{\text{ff}}) = O(8Ld^2)$
Feed-forward, MoE (active)	$O(2Lm d d_{\text{ff}}^{\text{exp}})$, i.e. $O(4Ld^2)$ for $m=2$, $d_{\text{ff}}^{\text{exp}}=d$	

TABLE III
SINGLE-SEED TEST ACCURACY (%) AT THE VALIDATION-LOSS-SELECTED MODEL STATE. PARAMETER TOTALS USE $C = 10$; CLASS-DEPENDENT HEADS ARE SLIGHTLY LARGER FOR CIFAR-100. **BOLD** MARKS THE LARGEST VALUE IN EACH DATASET COLUMN.

Model Variant	Steps	Embed.	Params (M)	CIFAR-10	CIFAR-100	MNIST	Fashion-MNIST
SA-MLP	8	Soft	13.13	92.0	68.5	98.2	91.1
SA-MoE	8	Soft	13.14	92.1	68.9	98.4	91.6
DSA-MoE	16	Hard	11.31	92.3	69.2	98.7	92.3

TABLE IV
EFFICIENCY AND RECURSION-DEPTH RESULTS ON CIFAR-10. PANEL (A) USES THE OPERATING POINTS IN TABLE III; BLOCK MULTIPLY-ADD RATIOS ARE EVALUATED AT $L = 10$ AND NORMALIZED TO SA-MLP. PANEL (B) REPORTS SINGLE-SEED ACCURACY; SA VARIANTS USE SOFT EMBEDDINGS, WHILE DSA-MoE USES HARD EMBEDDINGS, $k = 6$, $E = 4$, AND TOP-2 ROUTING.

(a) Operating-point efficiency			(b) Test accuracy (%) versus recursion depth					
Variant	MAC Ratio	Time (s/epoch)	Variant	4	6	8	12	16
SA-MLP	1.00×	180	SA-MLP	89.2	90.5	92.0	92.1	91.9
SA-MoE	0.67×	172	SA-MoE	89.6	90.9	92.1	92.2	92.1
DSA-MoE	0.52×	165	DSA-MoE	89.8	91.2	91.8	92.0	92.3

$N_{\text{step}} \in \{4, 6, 8, 12, 16\}$. The DSA ablation evaluates $k \in \{2, 4, 6, 8, L_t\}$, where $k = L_t$ denotes no pruning. Unless varied explicitly, MoE uses $E = 4$ experts, top- $m = 2$ routing, expert width d , and load-balancing coefficient $\lambda_{\text{load}} = 0.01$.

B. Selected Operating Points Across Datasets

Table III reports the selected operating point for each configuration. SA-MLP and SA-MoE use soft prediction embeddings and $N_{\text{step}} = 8$. DSA-MoE uses a hard embedding with the straight-through estimator, $N_{\text{step}} = 16$, and $k = 6$. The table therefore summarizes the best studied operating points rather than a fully matched causal comparison.

The selected DSA-MoE configuration has the largest single-seed accuracy on all four datasets, with margins of 0.2–0.7 percentage points over SA-MoE. Because it also uses a different recursion depth and embedding strategy, these margins should not be attributed to DSA alone. The ablations below separate the principal design choices more directly.

C. Efficiency and Recursion Depth

Table IV places measured training efficiency beside the recursion-depth ablation so that computational cost and accuracy saturation can be read together.

The theoretical block reductions are larger than the measured epoch-time reductions: SA-MoE is 4.4% faster than SA-MLP, and DSA-MoE is 8.3% faster. Wall-clock time also includes the convolutional encoder, optimization, routing and

TABLE V
EFFECT OF DSA BUDGET AT $N_{\text{STEP}} = 16$, WITH HARD EMBEDDINGS, $E = 4$, AND TOP-2 ROUTING. AVERAGE SPARSITY IS THE MEAN OF $1 - k'/L_t$ OVER ALL 16 STEPS; THE FULL ROW DISABLES PRUNING WHILE RETAINING THE DSA PROJECTIONS.

Budget k	CIFAR-10 Acc. (%)	Avg. Sparsity (steps 1–16)	Train Time (s/epoch)
Full ($k=L_t$)	92.0	0%	180
$k = 8$	92.2	24%	170
$k = 6$	92.3	36%	165
$k = 4$	91.4	52%	158
$k = 2$	89.1	75%	149

indexing overhead, and different recursion depths. All three variants improve from 4 to approximately 8–12 steps, after which the gains taper. DSA-MoE reaches its largest value at 16 steps, where $k = 6$ prunes tokens from steps 5 through 16; this pattern is consistent with selective use of a longer latent history but does not establish a causal advantage for sparsification.

D. Component Ablations

1) *DSA Sparsity Budget*: Table V varies the DSA budget k within DSA-MoE at $N_{\text{step}} = 16$ (so $L_{16} = 18$), reporting recursion-averaged sparsity alongside accuracy and training time.

A moderate budget of $k = 6$ gives the largest accuracy among the sparse settings. Reducing the budget to $k = 2$ increases average sparsity to 75% but lowers accuracy by 3.2 percentage points. The result illustrates the central trade-off:

TABLE VI

ROUTING AND EMBEDDING ABLATIONS ON CIFAR-10. PANEL (A) USES DSA ATTENTION WITH $k = 6$ AND HARD EMBEDDINGS; UTILIZATION CV IS THE COEFFICIENT OF VARIATION OF EXPERT TOKEN COUNTS. PANEL (B) COMPARES PREDICTION EMBEDDINGS; DSA-MoE USES $k = 6$, $E = 4$, AND TOP-2 ROUTING.

(a) MoE routing configuration				(b) Prediction embedding		
Configuration	Params (M)	Accuracy (%)	Util. CV	Variant	Embedding	Accuracy (%)
Dense MLP (no MoE)	11.29	92.0	N/A	SA-MLP	Soft	91.9
$E = 2$, top-1	7.10	91.6	0.15	SA-MoE	Soft	92.1
$E = 4$, top-1	11.31	92.0	0.22	DSA-MoE	Soft	91.8
$E = 4$, top-2	11.31	92.3	0.08	DSA-MoE	Hard (STE)	92.3
$E = 8$, top-2	19.72	92.4	0.18			

stronger pruning reduces training time but can remove latent context needed during later refinement steps.

2) *Routing and Embedding Strategy*: Table VI combines the MoE routing and prediction-embedding ablations at $N_{\text{step}} = 16$.

At $E = 4$, top-2 routing improves accuracy by 0.3 points over top-1 and has the lowest utilization coefficient of variation. Expanding from four to eight experts adds 74% more stored parameters for a further 0.1-point gain, making $E = 4$, top-2 the clearest balance among stored capacity, routing balance, and single-seed accuracy. Within DSA-MoE, the hard embedding improves accuracy by 0.5 points relative to the soft embedding; because hard embeddings were not evaluated for the standard-attention configurations, this remains a within-DSA comparison.

E. Internal Traces: What the Recurrence Actually Lets Us Observe

The recurrence exposes three complementary trajectories: DSA relevance scores over token roles, MoE routing weights over experts, and class probabilities derived from ℓ_t . Together they show how the model shifts attention among the image summary, the current prediction, and earlier latent states as refinement proceeds.

Token-role scope. Because global average pooling reduces the image to one vector before recursion, the tokens represent roles rather than spatial regions. The DSA scores can indicate increased or decreased reliance on x_{feat} , y_{t-1} , or a previous z_j , but they cannot identify a pixel, patch, edge, or object part. The appropriate interpretation is therefore a token-role interaction trace, not a spatial attention map.

Routing and confidence. Class-conditional expert frequencies can reveal associations between classes and routing decisions, but semantic labels such as “texture expert” require independent validation. Likewise, decreasing prediction entropy indicates increasing concentration, not necessarily increasing correctness or a faithful causal explanation. These trajectories are useful for describing model behavior while remaining distinct from formal interpretability or calibration guarantees.

VI. DISCUSSION

A. Accuracy–Efficiency Trade-off

The selected DSA-MoE operating point has the largest single-seed accuracy on all four datasets, but its margins are

small and its depth and embedding differ from the standard-attention operating points. The strongest supported conclusion is therefore that the three configurations achieve comparable accuracy, with DSA-MoE offering a favorable active-computation profile. At $L = 10$, its estimated transformer-block arithmetic is $0.52\times$ that of SA-MLP, while the measured epoch time is 8.3% lower. The difference between these ratios emphasizes that conditional arithmetic savings and realized hardware speed are distinct quantities.

B. Why the Growing-Sequence Design Matters

The decision to retain the latent history Z_t is the design choice that makes token sparsity meaningful. A fixed three-token $x/y/z$ recurrence would make any budget $k \geq 3$ dense at every step. The growing history instead gives later iterations access to the full refinement trajectory, while DSA limits how much of that trajectory is revisited. The resulting cost grows more slowly than dense attention as recursion deepens, but the benefit appears only after L_t exceeds k .

C. Interplay Between the Two Efficiency Mechanisms

DSA and MoE operate along different axes. DSA selects tokens from the growing history; MoE selects parameters for each token. Their analytical savings therefore accumulate rather than duplicate one another. The present ablations characterize each mechanism, but a complete 2×2 comparison of dense versus sparse attention and dense versus expert feed-forward layers across multiple seeds is still needed to estimate interaction effects.

D. Limitations

- **Statistical scope.** All reported results use one seed. Small differences, especially those below one percentage point, may reflect training variance.
- **Benchmark scale.** CIFAR-10/100, MNIST, and Fashion-MNIST are low-resolution classification datasets and do not establish behavior on larger images, more classes, or safety-critical domains.
- **Operating-point comparison.** The cross-dataset table uses the selected depth and embedding for each configuration. It should not be interpreted as a fully matched estimate of the isolated effect of DSA or MoE.
- **Spatial scope.** Global average pooling prevents region-level attribution. The internal traces describe token roles and routing decisions only.

- **Hardware dependence.** Top- k selection and expert routing introduce irregular memory access and dispatch, so measured speedups depend on the execution platform.
- **Fixed computation.** Every sample receives the same number of recursion steps, even though easy and ambiguous inputs may require different amounts of refinement.

VII. FUTURE DIRECTIONS

- **Model scaling.** Evaluating smaller embedding widths, narrower feed-forward layers, and fewer blocks would reveal the accuracy–parameter Pareto frontier of the recursive formulation.
- **Spatial tokens.** A compact patch-tokenized front end would allow DSA to select spatially meaningful regions while preserving a short sequence.
- **Differentiable selection.** Continuous relaxations of top- k , including sparse alternatives such as sparsemax [37], could provide smoother optimization and uncertainty over token relevance.
- **Intermediate supervision.** A weighted objective $\mathcal{L} = \sum_{t=1}^{N_{\text{step}}} w_t \mathcal{L}_{\text{CE}}(\ell_t, y^*)$ would test whether direct supervision of intermediate predictions improves refinement.
- **Adaptive halting.** A confidence- or state-stability criterion could allow easy examples to stop early and difficult examples to continue, following the principle of Adaptive Computation Time [25].
- **Statistical validation.** Multi-seed experiments and paired confidence intervals are required to distinguish the small observed differences from training variance.

VIII. CONCLUSION

This paper introduced a recursive visual classifier that refines a prediction through a growing history of latent states. The sequence length $L_t = t + 2$ links recursion depth directly to attention cost: DSA is dense at early steps and increasingly selective later, while MoE routing reduces the feed-forward parameters active for each token. The analytical treatment separates stored capacity, active parameters, transformer-block arithmetic, and measured training time.

Across CIFAR-10, CIFAR-100, MNIST, and FashionMNIST, the three studied configurations achieve comparable single-seed accuracy. The selected DSA-MoE operating point attains the largest value on each benchmark and the lowest estimated active computation, but the margins are small and the operating points are not fully matched. The results therefore motivate further study of recursive conditional computation rather than establishing a universal advantage. Multi-seed validation, spatial tokenization, and adaptive recursion are the clearest next steps.

AUTHOR CONTRIBUTIONS

The author conceived and designed the study, developed the methodology, conducted the experiments, analyzed the results, and prepared the manuscript.

ACKNOWLEDGMENTS

The author thanks the reviewers for their constructive comments on the presentation and evaluation of this work. The author also acknowledges the researchers behind the Hierarchical Reasoning Model and Tiny Recursive Model, whose studies provided the conceptual foundation for iterative refinement in this paper; the creators and maintainers of CIFAR-10, CIFAR-100, MNIST, and FashionMNIST for providing widely accessible evaluation benchmarks; and the PyTorch community for the open-source numerical and deep-learning tools that supported the experimental study.

DATA AVAILABILITY

The experiments use the publicly available CIFAR-10, CIFAR-100, MNIST, and FashionMNIST datasets cited in Section V-A. Training, validation, and test partitions follow the protocol stated there. The source code and experiment configurations are available at <https://github.com/codewithdark-git/VisionPulse>.

REFERENCES

- [1] F. Chollet *et al.*, “ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems,” *arXiv preprint arXiv:2505.11831*, 2025.
- [2] ARC Prize Foundation, “ARC Prize 2025 Results and Analysis,” *ARC Prize Blog*, 2025. [Online]. Available: <https://arcprize.org/blog/arc-prize-2025-results-analysis>.
- [3] J. Wei *et al.*, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 24824–24837.
- [4] C. Snell, J. Lee, K. Xu, and A. Kumar, “Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters,” *arXiv preprint arXiv:2408.03314*, 2024.
- [5] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, “Large Language Models are Zero-Shot Reasoners,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 22199–22213.
- [6] G. Wang, J. Li, Y. Sun, X. Chen, S. Liu, Y. Wu, Y. Lu, S. Song, and Y. Yang, “Hierarchical Reasoning Model,” *arXiv preprint arXiv:2506.21734*, 2025.
- [7] A. Jolicoeur-Martineau, “Less is More: Recursive Reasoning with Tiny Networks,” *arXiv preprint arXiv:2510.04871*, 2025.
- [8] F. Chollet *et al.*, “ARC Prize 2024: Technical Report,” *arXiv preprint arXiv:2412.04604*, 2024.
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.
- [10] DeepSeek-AI, “DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model,” *arXiv preprint arXiv:2405.04434*, 2024.
- [11] DeepSeek-AI, “DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models,” *arXiv preprint arXiv:2512.02556*, 2025.

- [12] Y. LeCun, C. Cortes, and C. J. C. Burges, "The MNIST Database of Handwritten Digits," 1998. [Online]. Available: <http://yann.lecun.com/exdb/mnist/>.
- [13] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
- [14] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Tech. Rep., University of Toronto, 2009.
- [15] 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," *arXiv preprint arXiv:1701.06538*, 2017.
- [16] A. Dosovitskiy *et al.*, "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale," in *Proc. Int. Conf. Learn. Represent.*, 2021.
- [17] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [18] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proc. Int. Conf. Mach. Learn.*, Long Beach, CA, USA, 2019, pp. 6105–6114.
- [19] J. L. Elman, "Finding Structure in Time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Apr. 1990.
- [20] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human Pose Estimation with Iterative Error Feedback," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 4733–4742.
- [21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [22] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," *arXiv preprint arXiv:2004.05150*, 2020.
- [23] M. Zaheer *et al.*, "Big Bird: Transformers for Longer Sequences," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 17283–17297.
- [24] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," *arXiv preprint arXiv:1904.10509*, 2019.
- [25] A. Graves, "Adaptive Computation Time for Recurrent Neural Networks," *arXiv preprint arXiv:1603.08983*, 2016.
- [26] N. Shazeer, "Fast Transformer Decoding: One Write-Head is All You Need," *arXiv preprint arXiv:1911.02150*, 2019.
- [27] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991.
- [28] D. Lepikhin *et al.*, "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding," *arXiv preprint arXiv:2006.16668*, 2020.
- [29] W. Fedus, B. Zoph, and N. Shazeer, "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," *arXiv preprint arXiv:2101.03961*, 2021.
- [30] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv preprint arXiv:1801.04381*, 2018.
- [31] Z. Liu *et al.*, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Montreal, QC, Canada, 2021, pp. 10012–10022.
- [32] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K., 2020, pp. 213–229.
- [33] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable Transformers for End-to-End Object Detection," in *Proc. Int. Conf. Learn. Represent.*, 2021.
- [34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.
- [35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 2818–2826.
- [36] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *arXiv preprint arXiv:1308.3432*, 2013.
- [37] A. F. T. Martins and R. F. Astudillo, "From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 1614–1623.