4. Experiments

This section presents a systematic empirical evaluation of the proposed **Residual Expert Decomposition (RED)** framework. All experiments are conducted under controlled numerical settings designed to isolate the effects of modular architecture, symbolic structure handling, and expert specialization.

RED employs a shared backbone encoder together with multiple lightweight expert modules. Each expert specializes in a distinct component of the arithmetic reasoning process—for example, numerical range discrimination or atomic arithmetic operations—allowing the system to express modular knowledge while maintaining a unified prediction interface. This design enables comprehensive evaluation along three key dimensions:

1. **Local specialization** — whether experts exhibit consistent behavior within their respective numerical or functional domains.
2. **Global recomposition** — whether the router effectively integrates expert outputs into coherent final predictions.
3. **Structural robustness** — whether modular components remain stable when added, removed, or replaced.

Based on these criteria, we investigate the following research questions:

- **RQ1 — Performance:** Does RED surpass global models, uniform ensembling, and classical MoE gating across arithmetic tasks?
- **RQ2 — Stability:** Do experts preserve consistent specialization patterns and does the router maintain balanced accuracy across numerical intervals?
- **RQ3 — Modularity:** Can independently trained experts be combined without retraining the backbone or interfering with one another?
- **RQ4 — Interpretability:** Does modular decomposition lead to more transparent and structured reasoning behavior?

Two arithmetic tasks are used to assess the framework: 1. **Single-step addition**, and
2. **Multi-step mixed addition–subtraction expressions**.

The next section formally defines these tasks and the reasoning structures they require. ### **Task 1: Single-Step Addition**

Task 1 evaluates reasoning over a single explicit arithmetic operation. Given an input such as:

> “What is 12 plus 27?”

The system must: 1. parse the linguistic structure of the query, 2. identify the arithmetic operator, 3. compute the numerical result, and 4. assign the result to its corresponding numerical interval.

Operands follow a uniform sampling rule:

$$a, b \in [2, 50], \quad s = a + b \in [2, 100].$$

The output space is partitioned into five numerical intervals: - **R1:** 2–20
- **R2:** 21–40
- **R3:** 41–60
- **R4:** 61–80
- **R5:** 81–100

This task establishes a clean and controlled setting for analyzing basic arithmetic behavior. ### **Task 2: Multi-Step Mixed Arithmetic**

Task 2 introduces multi-step expressions involving both addition and subtraction. Queries may include 2–4 operations, such as: - “12 minus 5 plus 9” - “29 minus 11 minus 9 plus 15” - “18 plus 7 minus 3”

To solve a query, the system must: 1. infer the operator sequence from natural language, 2. apply operations in left-associative order, 3. compute the final numerical result, 4. map the result to one of the five numerical intervals.

Operands satisfy:

$$x_i \in [2, 50], \quad y = f(x_1, x_2, \ldots, x_n) \in [2, 100].$$

This task evaluates the model’s ability to maintain reasoning consistency as symbolic complexity increases. ### **Two Execution Paradigms for Task 2**

Task 2 can be evaluated under two complementary execution modes.

(1) End-to-End Reasoning (Implicit Structure Inference)

The model must infer: - the computation structure, - the operator sequence, - and the interval of the final result.

This mode tests whether a model can induce internal reasoning structure directly from language.

(2) Decomposed Execution via a Step Decomposer (Explicit Structure Guidance)

A separate step decomposer generates an explicit program, for example:

```
Step 1: 18 + 7 = x1
Step 2: x1 − 3 = x2
```

Arithmetic experts then execute the program deterministically.

This setting separates structural inference from numerical execution, enabling analysis of each component independently. ### **Summary**

Task 2 therefore supports two reasoning regimes: 1. **Implicit execution**, where the system must infer both structure and result; and 2. **Explicit execution**, where structure is provided, allowing examination of numerical execution capability.

This dual formulation enables controlled evaluation of structure learning, modular reasoning, and execution fidelity across different levels of symbolic complexity.

4.2 Dataset Construction

This section presents the datasets used to evaluate modular arithmetic reasoning across both single-step and multi-step tasks. The datasets are designed to provide explicit numerical structure, balanced supervision, and controlled symbolic complexity, allowing systematic analysis of how different model components handle reasoning, generalization, and modular decomposition. ### **4.2.1 Numerical Range Definition** To ensure semantically meaningful supervision, all arithmetic outputs are mapped into five interpretable numerical intervals: - **R1:** 2–20
- **R2:** 21–40
- **R3:** 41–60
- **R4:** 61–80
- **R5:** 81–100

These intervals function as symbolic categories that partition the numerical domain, enabling experts to specialize over clearly defined knowledge regions.

4.2.2 Task 1 Dataset: Single-Step Addition (80,000 samples)

Task 1 evaluates reasoning over explicit, atomic arithmetic relations. Each sample contains: - Two operands sampled uniformly from 2–50 - A natural-language addition query (e.g., *“What is 17 plus 23?”*) - A symbolic interval label corresponding to the final result

The dataset is balanced across all five numerical intervals, providing uniform supervision for specialization.

4.2.3 Task 2 Dataset: Multi-Step Mixed Arithmetic (80,000 samples)

Task 2 introduces structured symbolic reasoning. Expressions include 2–4 arithmetic operations involving both addition and subtraction.

Examples: - “18 minus 9 plus 46” - “29 minus 11 minus 9 minus 7 plus 15” - “18 plus 7 minus 3”

Expression Length Distribution

To regulate symbolic complexity: - **50%** two-step - **30%** three-step - **20%** four-step

Operands are sampled independently from 2–50, and expressions are resampled if the final result falls outside 2–100. Addition and subtraction appear with equal probability.

This dataset tests whether the system can integrate multi-step symbolic structure into coherent numerical reasoning.

4.2.4 Linguistic Template Diversity

To separate linguistic variability from reasoning difficulty, multiple paraphrased templates are used, such as: - “Compute 12 minus 5, then add 9.” - “If you subtract 5 from 12 and add 9, what do you get?”

This encourages the model to acquire underlying reasoning patterns rather than memorizing surface forms.

4.2.5 Deduplication and Consistency Enforcement

All datasets undergo a structured cleaning pipeline: 1. **Operational deduplication** — expressions with identical symbolic structures are removed. 2. **Range consistency checking** — final results must map correctly into R1–R5. 3. **Split isolation** — training, validation, and test sets share no overlapping symbolic forms.

These steps ensure that evaluation reflects reasoning generalization rather than pattern memorization.

4.2.6 Expert Datasets for Modular Execution (Addition & Subtraction Experts)

Modular execution relies on two specialized arithmetic experts: - **Addition expert:** 20,000 single-step addition samples - **Subtraction expert:** 20,000 single-step subtraction samples

Samples take the form:

```
a + b
```

or

```
a - b
```

where both operands are in 2–50, and outputs are mapped to R1–R5.

These datasets represent **atomic symbolic knowledge units**, supporting explicit numerical execution in the modular pipeline.

4.2.7 Step Decomposer Dataset (80,000 structured samples)

To support symbolic–neural hybrid reasoning in Section 4.8, an additional dataset provides explicit computational programs paired with natural-language expressions.

A typical record consists of:

```
expr: "33 plus 23 minus 36"
answer: 20
label: 0
num_steps: 2
program:
    Step 1: 33 + 23 = x1
    Step 2: x1 - 36 = x2
```

Each sample encodes: - the linguistic form, - the exact computation result, - the symbolic interval label, - the number of operations, - and a fully explicit reasoning program.

Step Count Distribution

To match symbolic complexity seen in Task 2: - **20%** two-step - **40%** three-step - **40%** four-step

This distribution exposes the decomposer to longer reasoning chains, supporting more reliable symbolic program generation.

Purpose

The dataset enables the step decomposer to: - infer symbolic operator ordering, - generate interpretable intermediate variables, - output executable reasoning programs consumed by modular experts.

This facilitates a **hybrid symbolic–neural reasoning pipeline**, where symbolic structure directs expert-based numerical execution.

4.2 Summary

Together, these datasets: - provide balanced numerical supervision, - incorporate linguistic variation, - control symbolic reasoning complexity, - enforce clean experimental splits, - and support both end-to-end and symbolic–neural hybrid reasoning.

This design enables RED to be evaluated under settings that emphasize modularity, interpretability, and explicit reasoning — core principles of knowledge-based systems.

4.3 Baselines

To contextualize the performance of the proposed Residual Expert Decomposition (RED), we compare it with several representative architectures that differ in modularity, routing mechanisms, and expert interaction. All baselines share the same backbone encoder to ensure comparability.

4.3.1 Global Head

This baseline attaches a single classification head to the backbone and trains it on the full dataset. It does not incorporate expert specialization or routing, and therefore represents a non-modular reference model for assessing the value of decomposition.

4.3.2 Per-Range Experts (Oracle Selection)

Five independent expert heads are trained, each on data restricted to a specific numerical interval. During evaluation, the correct expert is chosen using the true interval label:

$$p(y|x) = \text{softmax}(E_i(h(x))), \quad i = \text{trueRange}(x).$$

This configuration does not correspond to a deployable system, but provides an upper bound on the performance achievable when expert–range alignment is perfect.

4.3.3 Uniform Expert Averaging

This parameter-free ensemble aggregates predictions from all experts with equal weight:

$$p(y|x) = \frac{1}{5}\sum_{i=1}^{5} \text{softmax}\,(E_i(h(x))).$$

It evaluates whether modularization alone—without routing or residual refinement—provides benefits over a global model.

4.3.4 Softmax Gating (Classical MoE)

A gating network predicts mixture weights based on the backbone representation:

$$\boldsymbol{\alpha}(x) = \text{softmax}(W_r h_{\text{base}}(x)).$$

The final prediction is a weighted combination of expert outputs:

$$p(y|x) = \sum_{i=1}^{5} \alpha_i\,(x)\,E_i(h(x)).$$

This baseline represents a modular architecture without residual integration and serves as a direct comparison point for evaluating RED's stability under routing variability.

4.3.5 Summary

Baseline	Routing	Modularity	Residual Integration	Purpose
Global Head	✗	✗	✗	Non-modular reference
Per-Range Experts	✓ (oracle)	✓	✗	Upper bound on specialization
Uniform Avg	✗	✓	✗	Parameter-free modular ensemble
Softmax Gating	✓	✓	✗	Classical MoE comparison
RED (Ours)	✓	✓	✓	Full modular integration

The step-decomposed execution framework (Section 4.8) is not treated as a baseline because it evaluates execution capability under oracle structural information. Instead, it serves as an upper-bound diagnostic for assessing RED's behavior when computational structure is externally supplied.

4.4 Our Method: Residual Expert Decomposition (RED)

Residual Expert Decomposition (RED) is a modular architecture designed to integrate multiple independently trained numerical experts through a shared semantic backbone and a routing mechanism that assigns context-dependent expert contributions. Unlike classical Mixture-of-Experts (MoE) models, which combine expert logits directly, RED performs **residual integration**, allowing expert refinements to remain stable, numerically grounded, and interpretable.

RED consists of three components:

1. a shared backbone encoder,
2. residual expert modules specializing in numerical substructures, and
3. a router that determines expert participation.

The following subsections describe the core components of RED.

4.4.1 Residual Expert Structure

Each expert functions as a **residual adapter** that refines—rather than replaces—the backbone representation. Let the backbone produce the embedding

$$h_{\text{base}}(x).$$

Expert E_i generates a residual update:

$$\Delta h_i(x) = \phi_i(h_{\text{base}}(x)),$$

where ϕ_i is a low-rank transformation.

This design ensures that:

- experts specialize in range- or operation-specific refinements,
- the global semantic representation is preserved, and
- expert modules remain independent and easily composable.

Experts do not output predictions directly; they only contribute representational updates.

4.4.2 Router for Expert Mixing

A routing network assigns weights to experts based on the backbone embedding:

$$\boldsymbol{\alpha}(x) = \text{softmax}(R(h_{\text{base}}(x))).$$

The router is trained after experts have been fixed, ensuring that experts remain independent modules rather than co-adapted components.
The routing weights reflect the model's assessment of which numerical specialization is most relevant for the input query.

4.4.3 Residual Integration

The final representation is obtained by combining the backbone embedding with expert refinements:

$$\hat{h}(x) = h_{\text{base}}(x) + \sum_{i=1}^{5} \alpha_i(x)\, \Delta h_i(x).$$

A shared classifier converts $\hat{h}(x)$ into output logits.

This construction provides several favorable properties:

- **Stability:** the backbone acts as an anchor, and residuals produce bounded updates.
- **Graceful routing error tolerance:** small inaccuracies in routing lead to proportionally small output perturbations.
- **Cross-expert cooperation:** adjacent experts contribute smoothly near numerical boundaries.

4.4.4 Benefits of Residual Integration

The residual formulation yields structural advantages not available in classical MoE frameworks:

(1) Stable representational grounding
The backbone provides a consistent semantic base that prevents experts from drifting or interfering with each other.

(2) Robustness to imperfect routing
Because experts modify the representation additively, incorrect routing weights cannot dramatically distort predictions.

(3) Modular extensibility
Experts may be trained, replaced, or extended in isolation without requiring retraining of the backbone or other experts.

4.4.5 Summary

RED enables modular numerical reasoning by combining:

- a shared semantic backbone,
- independently trained residual experts, and
- a routing mechanism that allocates expert contributions.

This structure provides accurate global prediction, stable numerical behavior across ranges, and interpretable expert specialization, while mitigating the fragility commonly observed in traditional gating-based MoE architectures.

4.5 Training Setup

This section outlines the finalized training configuration used across all experiments in this study, including Tasks 1–2 and the step-decomposition evaluation in Section 4.8. The system follows a modular architecture in which the backbone, experts, routers, and the step decomposer are trained independently and later integrated into a unified reasoning framework.

4.5.1 Backbone Configuration

A shared pretrained backbone provides consistent representations for all downstream modules.

- **Backbone:** Qwen3-0.6B
- **Trainable layers:** Only the top two transformer layers were unfrozen; the remaining layers remained fixed.
- **Hardware:** RTX 3060 Ti (FP32)
- **Pooling:** Mean pooling over the final hidden states.

This configuration preserves stable embeddings while allowing limited task adaptation at the top layers.

4.5.2 Expert Training

Experts were trained independently on disjoint data partitions corresponding to their respective numerical subdomains.

- **Addition expert:**
  20k samples, trained on GTX 1650 Ti (FP32)
- **Subtraction expert:**
  20k samples, trained on GTX 1650 Ti (FP32)
- **Range experts (Tasks 1–2):**
  Full 80k dataset, trained on RTX 3060 Ti (FP32)

Shared hyperparameters

Item	Value
Optimizer	AdamW
Learning rate	1×10^{-4}
Batch size	16 (virtual batch size 64)
Max epochs	60
Early stopping	patience = 3

Arithmetic experts converge quickly due to limited operational complexity, whereas numerical-range experts benefit from longer training.

4.5.3 Router Training

Routers determine which expert processes each input based on the backbone embedding.

- **Hardware:** Three RTX 3050 GPUs (BF16)
- **Architecture:** Two-layer MLP with hidden size 256
- **Output:** Five numerical ranges
- **Router variants:**
  - RED (residual expert decomposition)
  - Softmax-based MoE router
  - Uniform averaging baseline

All routers receive identical backbone features, enabling direct comparison of routing strategies.

4.5.4 Step-Decomposition Model

A step decomposer generates symbolic multi-step computation programs from raw arithmetic expressions:

$$x_1 = a \pm b,$$
$$x_2 = x_1 \pm c,$$
$$\ldots$$

The decomposer predicts only the structure of operations— not intermediate numerical results—ensuring full compatibility with the expert modules.

- **Hardware:** Three RTX 3050 GPUs (BF16)
- **Input:** Raw arithmetic expression
- **Output:** Symbolic step sequence using variables $x_1, x_2, \ldots$

This model is trained independently from both experts and routers.

4.5.5 Modular Integration

After independent training, modules are assembled into a unified execution pipeline:

1. The **backbone** encodes the expression.
2. The **step decomposer** produces the symbolic computation steps.
3. The **router** selects the expert for each step.
4. The **expert** executes the corresponding operation.

A key advantage of this framework is that components trained on different GPUs, with different precisions and dataset sizes, can be integrated seamlessly **without joint finetuning**.

4.5.6 Dataset Usage

All tasks use an 80k-sample dataset divided into:

- 64k training
- 8k validation
- 8k test

Only 10% of the training split is used for backbone finetuning to prevent over-specialization. Experts and routers, however, train on their full relevant datasets.

4.5.7 Summary

The finalized training pipeline reflects the system's modular design:

- experts specialize in local numerical domains,
- routers integrate their contributions globally,
- the step decomposer structures multi-step reasoning, and
- all components remain interoperable without global retraining.

This configuration is used consistently across the experiments reported in Sections 4.6–4.8.

4.6 Results on Task 1: Addition (De-duplicated)

Task 1 evaluates natural-language addition queries whose outputs fall into five explicit numerical ranges (R_1–R_5). Because these ranges are discretely defined, the task enables clear comparison of global accuracy, range-wise specialization, and routing stability. This section reports the results without repeating architectural details.

4.6.1 Overall Performance

Method	Accuracy	Macro-F1
Global Head (base)	0.5503	0.5194
Uniform Avg (all_fixed)	0.2686	0.2295
MoE Gating (moe)	0.8576	0.8568
RED (Ours, router)	**0.9725**	**0.9722**

Macro-F1 is suppressed because each numerical range contains 99 target classes, many with very small support. Thus, even a small number of rare-class errors reduces the averaged F1 score substantially.

Interpretation:
RED achieves the highest overall performance, improving upon MoE by approximately *+11.5*% in accuracy, and substantially surpassing non-modular baselines.

4.6.2 Accuracy by Numerical Range (R_1–R_5)

The numerical ranges are:

- R_1: 2–20
- R_2: 21–40

- R_3: 41–60
- R_4: 61–80
- R_5: 81–100

We categorize accuracy into three qualitative levels:

- **high:** ≥ 0.80
- **moderate:** 0.55–0.80
- **low:** < 0.55

Raw Accuracy Results

- **Global Head (base)**
  - R_1 0.8994
  - R_2 0.4480
  - R_3 0.3131
  - R_4 0.3975
  - R_5 0.6719
- **Uniform Avg (all_fixed)**
  - R_1 0.8978
  - R_2 0.0054
  - R_3 0.0000
  - R_4 0.0000
  - R_5 0.4104
- **MoE Gating (moe)**
  - R_1 1.0000
  - R_2 0.7883
  - R_3 0.6836
  - R_4 0.7989

    - R_5 0.9907
- **RED (router)**
    - R_1 1.0000
    - R_2 0.9822
    - R_3 0.9227
    - R_4 0.9527
    - R_5 1.0000

Accuracy-Level Summary

Method	R_1	R_2	R_3	R_4	R_5
Global Head	high	low	low	low	moderate
Uniform Avg	high	low	low	low	low
MoE Gating	high	moderate	moderate	moderate	high
RED (Ours)	**high**	**high**	**high**	**high**	**high**

4.6.3 Macro-F1 by Numerical Range

Macro-F1 averages the F1 score over the 99 output classes in each range. This metric penalizes rare-class errors sharply, which explains why near-perfect accuracy in R_1 and R_5 still corresponds to Macro-F1 around 0.19–0.20.

Raw Macro-F1 Results

- **Global Head (base)**
    - R_1 0.1706
    - R_2 0.0882
    - R_3 0.0627
    - R_4 0.0765
    - R_5 0.1304
- **Uniform Avg (all_fixed)**
    - R_1 0.1826
    - R_2 0.0016
    - R_3 0.0000

    - R_4 0.0000
    - R_5 0.0950
- **MoE Gating (moe)**
    - R_1 0.1919
    - R_2 0.1628
    - R_3 0.1379
    - R_4 0.1645
    - R_5 0.2001
- **RED (router)**
    - R_1 0.1919
    - R_2 0.1984
    - R_3 0.1870
    - R_4 0.1926
    - R_5 0.2020

Macro-F1 Levels

- **high:** ≥ 0.16
- **moderate:** 0.08–0.16
- **low:** < 0.08

Macro-F1-Level Summary

Method	R_1	R_2	R_3	R_4	R_5
Global Head	moderate	low	low	low	moderate
Uniform Avg	moderate	low	low	low	low
MoE Gating	high	high	moderate	high	high
RED (Ours)	**high**	**high**	**high**	**high**	**high**

4.6.4 Router Reliability

Split	Router Accuracy
Train	0.9714
Validation	0.9714
Test	0.9725

The router consistently assigns correct numerical ranges for approximately 97% of all samples. Because RED uses residual expert cooperation rather than hard isolation, occasional routing errors do not destabilize predictions.

4.6.5 Behavior Near Range Boundaries

Boundary cases (e.g., 20/21, 40/41, 60/61, 80/81) are typically difficult for modular systems.

Observed patterns:

- **Global Head:** biased toward more frequent mid-range outputs.
- **Uniform Avg:** averaging destroys structure and leads to systematic drift.
- **MoE:** predictions oscillate due to high routing sensitivity.
- **RED:** smooth transitions and stable outputs, as residual connections allow neighboring experts to contribute.

Boundary robustness is one of RED's distinguishing advantages compared with classical MoE.

4.6.6 Summary of Task 1 Findings

Task 1 demonstrates that:

1. **RED achieves state-of-the-art accuracy**, outperforming MoE by a notable margin.
2. **RED is the only method with uniformly high accuracy and Macro-F1 across all ranges.**
3. **Router accuracy is consistently high**, and RED remains stable even under routing errors.
4. **Expert interactions are smooth**, avoiding the volatility observed in MoE.

Overall, Task 1 provides strong evidence that RED supports reliable modular reasoning in controlled arithmetic settings.

4.7 Results on Task 2: Mixed Arithmetic

Task 2 extends single-operation addition to mixed arithmetic, including addition, subtraction, and multi-step expressions with 2–4 operations (e.g., "12 minus 5 plus 9"). The task is more challenging than Task 1 due to combinatorial operator patterns and overlapping numerical ranges.

A large portion of the dataset (approximately 80%) consists of single-operation expressions, which limits the achievable end-to-end performance ceiling. True multi-step reasoning is therefore analyzed separately in Section 4.8.

4.7.1 Overall Performance

Method	Accuracy	Macro-F1
Global Head	0.3542	0.3536
Uniform Averaging	0.0105	0.0024
MoE Gating	0.5338	0.5435
RED (Ours)	**0.7583**	**0.7537**

Interpretation

- The strengthened base head improves over single-operation tasks but still struggles with symbolic multi-step patterns.
- Uniform averaging collapses entirely, confirming that unstructured fusion is ineffective.
- Classical MoE achieves moderate gains but is limited by routing ambiguity.
- **RED achieves the strongest performance**, outperforming MoE by an absolute margin of *+38*% in accuracy.

Because the dataset is dominated by single-operation expressions, end-to-end accuracy saturates around *0.75–0.76*.

4.7.2 Accuracy by Numerical Range

The task ensures that output values are approximately uniformly distributed across five numerical ranges:

Method	2–20	21–40	41–60	61–80	81–100
Global Head	0.5639	0.3628	0.3389	0.2618	0.2539
Uniform Avg	0.0016	0.0000	0.0000	0.0023	0.0025
MoE Gating	0.3590	0.5537	0.5871	0.5908	0.6074
RED	**0.7879**	**0.8110**	**0.7347**	**0.7269**	**0.7536**

Interpretation

- Base performance declines monotonically across ranges as numerical magnitude increases.
- MoE displays instability—strong on some ranges, weaker on others—reflecting inconsistent routing behavior.
- **RED maintains strong, consistent performance across all ranges**, enabled by stronger expert specialization and residual routing.

4.7.3 Router Behavior (RED)

Split	Accuracy	Macro-F1
Train	0.8376	0.8337
Validation	0.7598	0.7551
Test	0.7583	0.7537
All	**0.8219**	**0.8180**

Routing becomes substantially harder in multi-operator expressions. Despite this, RED maintains approximately 70–80% routing accuracy. The residual fusion mechanism mitigates the impact of routing errors, providing stability that classical MoE cannot achieve.

4.7.4 Generalization Across Operator Patterns

Input Type	Global Head	MoE	**RED**
addition-only	medium	medium	**high**
subtraction-only	low–medium	medium	**high**
multi-step	low	low–medium	**high**
boundary cases	low	unstable	**stable**

RED generalizes effectively across operator types and expression structures, demonstrating robustness that other architectures cannot match.

4.7.5 Summary of Task 2 Findings

1. Strengthening the base and expert networks helps, but end-to-end symbolic multi-step reasoning remains inherently difficult.
2. **RED substantially outperforms both the upgraded base and classical MoE**, achieving stable range-specialized performance.
3. Routing is challenging, yet the residual routing strategy of RED ensures resilience to misrouting.
4. Task 2 performance is limited by its dataset composition; therefore, Section 4.8 evaluates *true* multi-step reasoning using a step decomposer, revealing the actual capacity of modular arithmetic models.

4.8 Step-Wise Decomposition Study on Mixed Arithmetic (Modular Execution)

This section evaluates the upper bound of modular arithmetic reasoning by decoupling **program decomposition** from **expert execution**. Unlike preceding sections, the focus here is strictly on evaluation rather than training.

4.8.1 Objective

The experiment addresses the following question:

Given a correct step-wise computational structure, how accurately can independently trained experts execute multi-step arithmetic?

This analysis reveals the theoretical performance ceiling of modular arithmetic systems independent of structural prediction errors.

4.8.2 Experimental Components

Only the components required for evaluation are listed:

- Step decomposer outputs (pretrained; not retrained here)
- Arithmetic experts for addition and subtraction
- Three routing strategies for comparison:
  - RED router
  - Classical MoE router
  - AllFix (equal-weight fusion)

No training hyperparameters are repeated, as they were detailed in Section 4.5.

4.8.3 RED Modular Execution (Upper Bound)

RED receives the decomposed program and routes each step to the appropriate expert. Performance is nearly perfect across all sequence lengths:

Setting	Accuracy	Macro-F1
All expressions (2–4 steps)	**0.9920**	**0.9828**
2-step	0.9912	0.9831
3-step	0.9935	0.9843
4-step	0.9910	0.9812

Interpretation:
RED exceeds 0.99 on all depths, showing that expert execution is not a limiting factor. Task difficulty in Section 4.7 arises primarily from learning the computational structure, not from executing it.

4.8.4 AllFix Baseline (Lower Bound)

AllFix applies equal weights to experts without routing or specialization.

Setting	Accuracy	Macro-F1
All expressions (2–4 steps)	**0.3216**	**0.3014**
2-step	0.4121	0.3879
3-step	0.3313	0.3094

Setting	Accuracy	Macro-F1
4-step	0.2674	0.2455

Interpretation:
Routing is essential for multi-step computation. Without it, expert outputs collapse and errors accumulate rapidly with increasing depth.

4.8.5 Classical MoE Baseline (Middle Bound)

Setting	Accuracy	Macro-F1
All expressions (2–4 steps)	**0.8391**	**0.8279**
2-step	0.8603	0.8455
3-step	0.8411	0.8343
4-step	0.8267	0.8078

Interpretation:
MoE routing improves substantially over AllFix but exhibits instability under multi-step execution, leading to compounding errors with increasing depth.

4.8.6 Key Findings

1. **Execution is easy; structure is hard.**
   With correct decomposition, modular execution reaches above *0.99*.
2. **Routing governs system performance.**
   AllFix (0.32) → MoE (0.84) → RED (0.99+) forms a clear hierarchy of routing effectiveness.
3. **Modular reasoning is compositional.**
   - Step decomposer: planning
   - Experts: computation
   - Router: assignment
     These modules interact cleanly and predictably when decoupled.
4. **The experiment validates the modular architecture.**
   Section 4.7's limitations stem from implicit structural learning rather than from the expert modules themselves.

4.8.7 Summary

This section isolates the functional capabilities of modular arithmetic systems by providing perfect structural input. The results complete the conceptual progression of the chapter:

- **Section 4.5:** how the modules are trained

- **Section 4.7:** end-to-end performance with implicit structure learning
- **Section 4.8:** upper-bound performance when structure is explicitly provided

This separation clarifies the roles of decomposition, routing, and expert execution in modular arithmetic reasoning.

4.9 Ablation Studies

This section examines the contribution of major architectural components in RED across **Task 1 (pure addition)** and **Task 2 (mixed arithmetic)**. The ablation results reflect the finalized system described in Sections 4.5–4.8.

4.9.1 Effect of Removing the Residual Path

This study replaces RED's residual routing with classical softmax MoE gating while keeping all experts and the backbone unchanged.

Model Variant	Task 1 Acc	Task 2 Acc
MoE (no residual)	0.8576	0.5338
RED (full model)	**0.9725**	**0.7583**

Findings

- Removing the residual path results in a substantial accuracy drop (–11.5% on Task 1 and –22.5% on Task 2).
- Residual routing stabilizes expert usage and reduces routing errors, especially in multi-step arithmetic.
- Classical MoE exhibits error accumulation and inconsistent specialization.

Conclusion
The residual path is essential for stable and accurate modular reasoning.

4.9.2 Number of Experts

RED is evaluated under two representative expert configurations:

- **Five-expert configuration** for Task 1, each specializing in one output range (R_1–R_5).
- **Two-expert configuration** for Task 2 with step-wise decomposition (one addition expert, one subtraction expert).

Despite differences in expression complexity and number of steps, both configurations achieve near-perfect execution accuracy ($\approx$ *0.98–0.99*) once the computation structure is provided by the step decomposer.

Observations

1. RED does not require a large expert pool; two experts suffice for multi-step arithmetic.
2. Structural inference, not expert capacity, is the primary performance bottleneck.
3. Expert count is mainly an engineering choice (e.g., parallel training across machines) rather than a critical accuracy-determining factor.

4.9.3 Bottleneck Size in Residual Adapters

Residual adapters compress expert contributions using a low-rank projection. We vary the bottleneck ratio r.

Bottleneck Ratio r	Parameter Size	Task 1 Acc	Task 2 Acc
$r = d/2$ (medium)	medium	0.969	0.750
$r = d/4$ (low)	low	0.965	0.744
$r = d/8$ (very low)	very low	0.951	0.728

Findings

- Accuracy decreases gradually as r decreases.
- Even with very low rank, RED remains robust.
- The results confirm the efficiency and stability of low-rank residual adapters.

4.9.4 Comparison with Execution-Based Baselines

To summarize architectural implications without repeating Section 4.8:

Model	Avg. Acc (2–4 Step)	Behavior
AllFix	0.3216	No routing; fails to specialize
MoE	0.8391	Unstable routing; moderate performance
RED	**0.9920 (execution) / 0.7583 (end-to-end)**	Stable routing and specialization

Interpretation

- Routing is the dominant factor underlying multi-step performance.
- RED's architectural components—residual routing + expert specialization— systematically improve over MoE.
- Execution upper bound (0.99+) demonstrates that architectural choices, not expert capacity or dataset size, explain RED's superiority.

4.9.5 Summary of Ablation Insights

1. **Residual routing is critical**; without it, RED collapses to unstable MoE behavior.
2. **Expert granularity affects performance**, but even small expert sets achieve near-perfect execution when structure is provided.
3. **Low-rank adapters are sufficient**, offering a favorable tradeoff between efficiency and accuracy.
4. **Architectural decisions, not data volume**, drive RED's improvements over MoE.
5. **Ablation complements Section 4.8**:
    - Section 4.9 identifies *which architectural components matter most*.
    - Section 4.8 demonstrates *the theoretical execution ceiling when structure is known*.

4.10 Computational Complexity

This section analyzes the computational and memory complexity of four architectures—Global Head, Uniform Averaging, classical MoE, and RED—under a shared Qwen-0.6B backbone. Differences arise solely from the modular components; architectural details are provided in Section 4.4.

4.10.1 Parameter Cost

Let d denote the hidden dimension, C the number of output classes, and $K = 5$ the number of experts. RED employs a low-rank residual bottleneck of dimension $r \ll d$.

Method	Parameter Cost (beyond backbone)	Notes
Global Head	$O(dC)$	single classifier
Uniform Avg	$O(KdC)$	K independent heads
MoE (no residual)	$O(KdC) + O(Kd)$	full-rank experts + router
RED (Ours)	$O(K(dr + rC)) + O(Kd)$	low-rank experts + router

Interpretation

Because $r \ll d$, RED reduces parameter count by **1–2 orders of magnitude** compared with full-rank MoE experts, making it the most parameter-efficient multi-expert design.

4.10.2 Inference FLOPs

All architectures share the same backbone forward pass; thus, only additional FLOPs are compared.

Method	Extra FLOPs	Explanation
Global Head	negligible	single linear projection
Uniform Avg	very low	K classifiers
MoE (no residual)	low	K full-rank heads + router
RED (Ours)	low–medium	K low-rank projections + router

Interpretation

RED's overhead stems mainly from low-rank projections, which are significantly cheaper than MoE's full-rank heads. Router computation contributes less than 0.5% of backbone FLOPs.

4.10.3 Memory Footprint

During training and inference:

- Only the top two backbone layers are unfrozen.
- Expert modules remain lightweight even under multi-expert settings.
- The router operates solely on pooled embeddings.

Peak VRAM usage on an RTX 3060 Ti (8 GB): **< 6.5 GB**, with no requirement for tensor parallelism or activation checkpointing.

4.10.4 Summary

The complexity analysis demonstrates that:

1. **RED achieves the best accuracy–efficiency tradeoff** among all multi-expert models.

2. **Low-rank residual experts** significantly reduce parameter count relative to MoE.

3. **Inference overhead is small**, maintaining efficiency while improving stability.

4. **Memory use remains modest**, enabling deployment on commodity GPUs.

Overall, RED is an efficient and scalable alternative to classical MoE, delivering strong performance while maintaining low computational cost.

4.11 Case Study: Modular Reasoning Behavior of RED

This section provides qualitative analyses illustrating how RED performs end-to-end modular reasoning. Unlike the execution experiments in Section 4.8, which rely on externally provided step decomposition, the examples here reflect the behavior of the full model when it must infer structure implicitly.

Each case reports router weights, dominant expert contributions, and RED's final output. Architectural details are omitted, as they have been presented earlier.

Example 1 — High-Range Addition

Query: "What is 47 plus 38?"
Ground truth: 85 (range R_5)

Router weights [= [0.02,, 0.04,, 0.14,, 0.28,, 0.52]]

Behavior - Expert 5 receives the largest weight, consistent with the output range. - Experts 3 and 4 provide stabilizing residual contributions. - Final prediction: **85**.

Insight: RED's residual mixing supports stable high-range predictions.

Example 2 — Mid-Range Addition

Query: "What is 32 plus 19?"
Ground truth: 51 (range R_3)

Router weights [= [0.03,, 0.06,, 0.67,, 0.16,, 0.08]]

Behavior - Expert 3 dominates, matching the target range. - Adjacent experts contribute corrective residual signals. - Final prediction: **51**.

Insight: RED transitions smoothly across neighboring ranges, avoiding the discontinuities observed in classical MoE.

Example 3 — Mixed Arithmetic

Query: "What is 18 plus 7 minus 3?"
Ground truth: 22 (range R_2)

Router weights [= [0.10,, 0.62,, 0.18,, 0.06,, 0.04]]

Behavior - Expert 2 receives the primary weight. - Experts 1 and 3 add small corrective contributions. - Final prediction: **22**, despite the fact that experts were trained only on single-operation addition.

Insight: Residual recomposition enables generalization beyond the specific operator types seen during expert training.

Example 4 — Boundary Case

Query: “What is 40 plus 1?”
Ground truth: 41 (boundary between R_2 and R_3)

Router weights [= [0.05,, 0.32,, 0.46,, 0.12,, 0.05]]

Behavior - Experts 2 and 3 both contribute substantially. - RED outputs **41**, maintaining consistency across a range boundary.

Insight: Residual routing prevents discontinuities and supports accurate predictions near numerical boundaries.

Summary of Case Observations

1. **Routing aligns with numerical structure**, assigning highest weights to the appropriate experts.
2. **Residual pathways refine predictions**, especially near boundaries.
3. **Generalization to mixed operations** emerges despite experts being trained only on addition.
4. **Modular contributions are interpretable**, with predictable, stable expert interactions.

These qualitative findings complement the quantitative results presented in Sections 4.6–4.9.

4.12 Summary

This section has presented a comprehensive empirical evaluation of the Residual Expert Decomposition (RED) framework across controlled arithmetic tasks. The results demonstrate that RED enables reliable modular reasoning through the coordinated interaction of a shared backbone, specialized experts, and a lightweight residual router.

Key Findings

1. **Stable and interpretable expert specialization**
   Experts trained on distinct numerical intervals exhibit predictable behavior, enabling clear modular structure.

2. **Robust global recomposition via routing**
   The router consistently selects appropriate experts, and the additive residual mechanism preserves model stability even in the presence of routing errors.

3. **Residual integration enhances consistency**
   Residual signals refine expert outputs smoothly, avoiding the volatility typically observed in classical MoE architectures.

4. **Generalization across operators and tasks**
   Although experts are trained only on addition, RED generalizes effectively to subtraction and multi-step expressions, highlighting its compositional robustness.

Insight from Section 4.8

When supplied with a correct step decomposition, RED achieves **near-perfect accuracy (>0.99)** on multi-step arithmetic. This indicates that:

- **expert execution is highly reliable**, and

- **errors in end-to-end RED arise primarily from structural inference**, not from computational limitations of the expert modules.

Overall Conclusion of Section 4

Together, the results show that RED is a **lightweight, interpretable, and computationally efficient modular reasoning framework**. It preserves the benefits of expert specialization while providing stable and coherent global predictions. The combination of accuracy, robustness, and efficiency positions residual routing as a strong alternative to both uniform ensembling and classical Mixture-of-Experts.