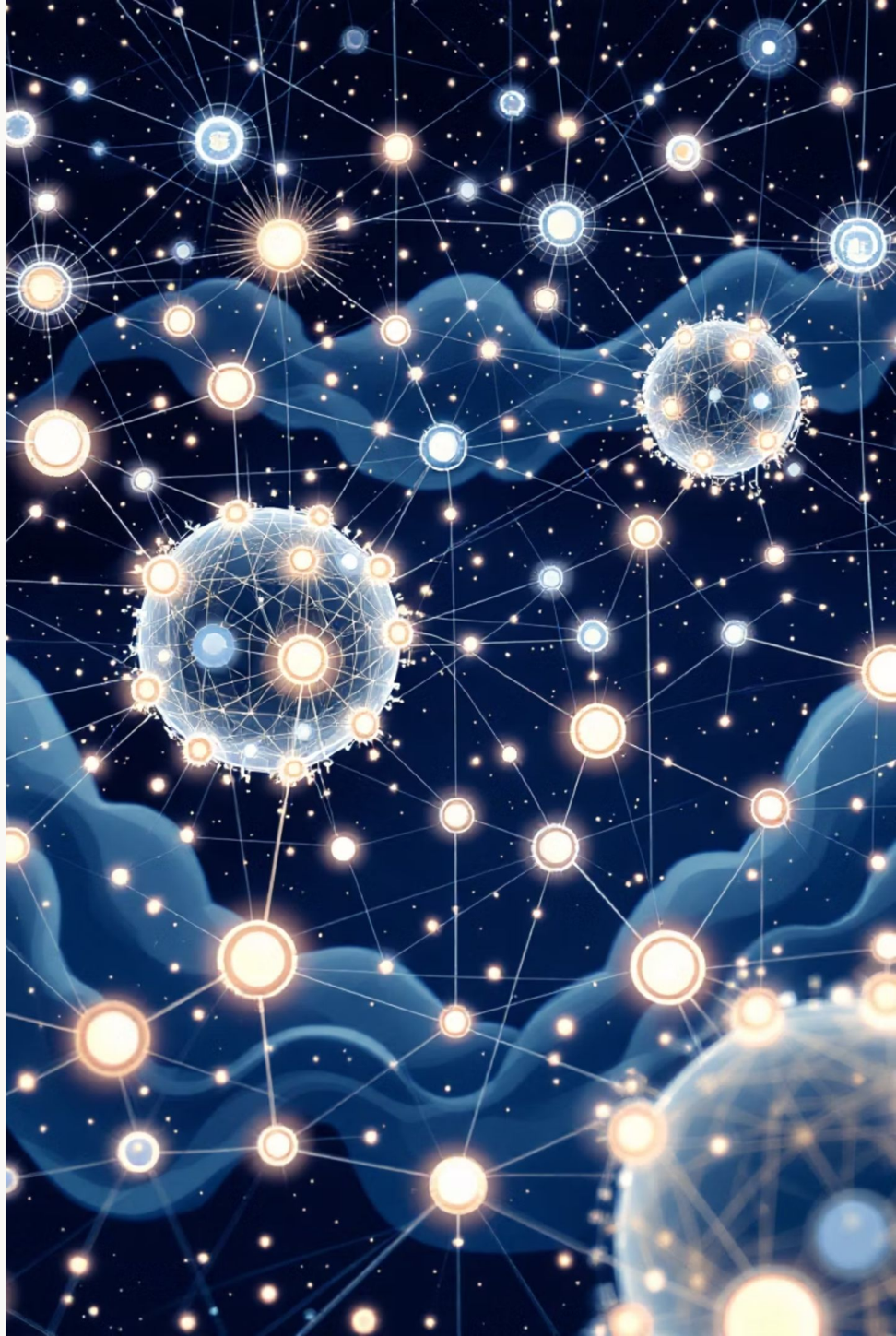
Efficient Dynamic Ensembling for Multiple LLM Experts

A novel approach to combining the strengths of multiple Large Language Models through intelligent sequential reasoning and knowledge transfer.



Understanding LLM Ensembling

Why Combine Multiple Experts?

Ensembling is a strategy for combining multiple LLMs to leverage their individual strengths. Instead of relying on a single model, multiple LLM "experts" collaborate to produce better, more consistent results across diverse tasks.

Different Strengths

Each LLM differs in architecture, size, and training data—making them strong in some areas but weak in others.

Better Generalization

Combining them helps achieve more robust performance across diverse tasks and domains.

Improved Efficiency

A well-designed ensemble can improve accuracy while reducing total computation cost compared to running one large model.

Four Existing Types of Ensemble Methods

Existing LLM ensemble approaches fall into distinct categories, each with unique strengths and limitations:

1

Mixture-of-Experts (MoE)

Uses a router network to select and activate a subset of expert models. Aggregates diverse expertise but requires retraining and cannot combine non-homologous LLMs or exploit complementary knowledge.

2

Parameter Merging

Merges parameters of multiple homologous LLMs into a single unified model. Methods like Fisher Merging and DARE work only for models with same initialization and structure.

3

Rule-Based Methods

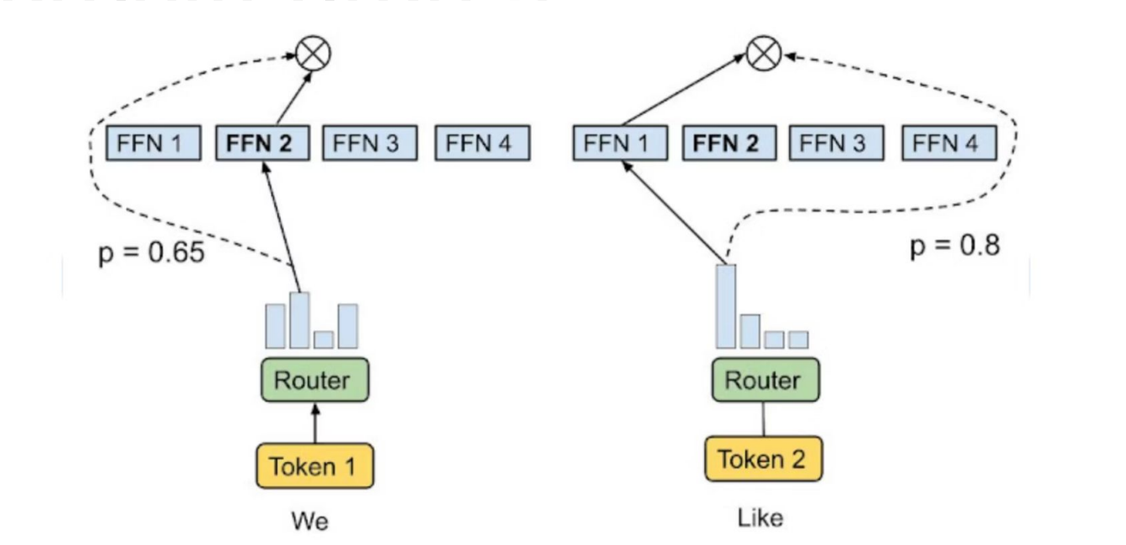
Combines LLMs through manually designed roles or fixed sequences. Simple to implement but static setup provides poor generalization to new domains or tasks.

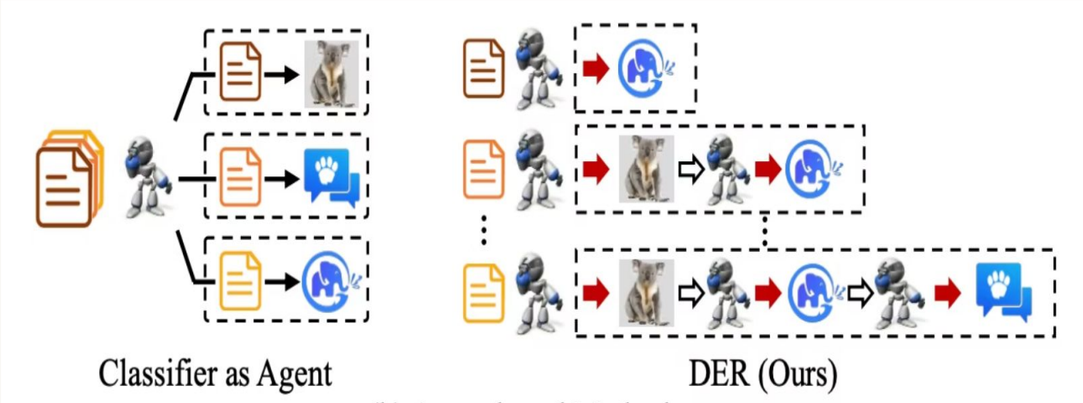
4

Agent-Based Methods

Trains an agent to dynamically select LLMs based on input. Can integrate non-homologous models but doesn't fully utilize complementary knowledge between experts.

A high level overview of different architectures





The Core Challenge

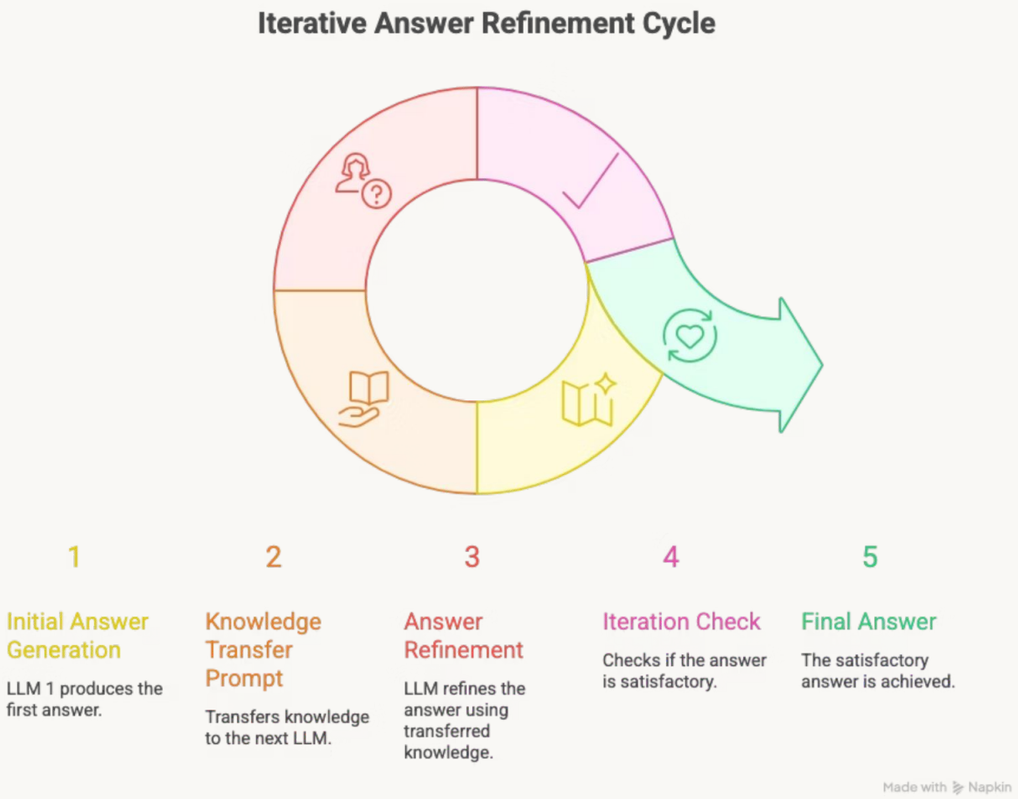
Existing ensemble methods face critical limitations that prevent optimal performance:

Computational Inefficiency

Most methods require running all LLM candidates simultaneously, wasting resources on easy samples that could be solved by a single model. For example, Pair-Ranker requires over 117B parameters in inference, leading to significant resource wastage.

Limited Knowledge Leverage

Agent-based methods that select only one LLM per decision have minimal inference cost but fail to leverage complementary knowledge among experts, limiting overall performance quality.



The algorithm for DER:

We seek a general DER Agent to select an optimal sequential route obtaining **the highest performance** and **lowest computational resources**

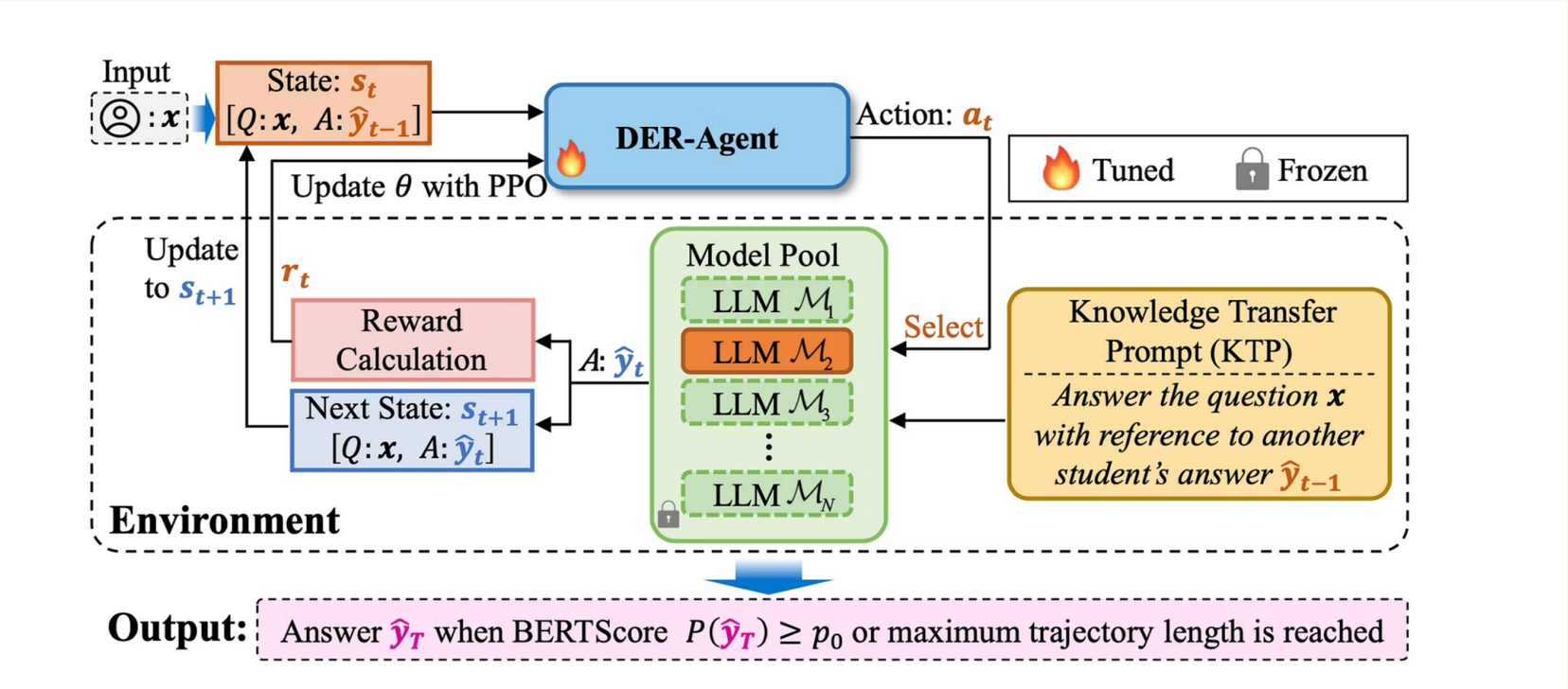
Initialise: Agent receives question with no prior answer

Select LLM: DER-Agent chooses next expert based on question and current answer state.

Transfer Knowledge: Knowledge Transfer Prompt enables new LLM to build on previous answer.

Evaluate: Reward function assesses quality improvement and computational cost.

Terminate or continue: Process stops when answer quality threshold reached or max steps exceeded.



At time step t, the agent takes the state s_t as input and performs an action a_t through the policy network $\pi: S \times A \rightarrow [0, 1]$. The environment changes to the next state $s_{t+1} = T(s_t, a_t)$ according to the transition function T, and a reward $R_t = R(s_t, a_t)$ is received with the reward function R.

MDP Architecture: How It Works

The system operates through five key components working together:

States

Question-answer pairs: [Q: input, A: previous answer] describing the current environment and quality of answers.

Actions

Discrete selection of which LLM expert to activate next from the available model pool.

Transitions

Is a function to map a state into a new state. Selected LLM generates new answer using Knowledge Transfer Prompt, creating new state with improved response.

Rewards

In the LLM ensemble task, the reward can be considered as an evaluation of the quality of the answer y for the selected LLM Ma.

Policy

Neural network that outputs probability distribution over LLM actions based on current state.

$$\pi(a_t = i \mid s_t; \theta) = \frac{\exp f_\theta(s_t)_i}{\sum_{j=1}^{N} \exp f_\theta(s_t)_j}$$

The Dynamic Reward Function for DER

The reward function is crucial for guiding the DER agent, balancing 3 things: **the quality of generated answers, the increment quality of the answer, and the computational resources**. It adapts based on the current step in the expert selection process.

$$R_t = \begin{cases} P(\hat{y}_t) - \alpha C(M_a) & \text{if } t = 0 \\ P(\hat{y}_t) + \beta \Delta P(\hat{y}) - \alpha C(M_a) & \text{if } t > 0 \end{cases}$$

Where:

P(\hat{y}_t): BERTScore, evaluating the quality of the answer from the selected LLM *M_a*.

C(M_a): Computational cost of activating the LLM *M_a*.

\Delta P(\hat{y}): Increment in BERTScore from the previous step *(t-1)* to the current step *(t)*.

\alpha, \beta: Coefficients to tune the influence of computational cost and quality increment.

An additional term encourages efficient termination of the process:

$$R(s_t, a_t) = \quad R(s_t, a_t) + \quad \gamma \quad \text{if } P(\hat{y}_t) \geq \ p_0 \text{ and } t \leq T_{max} \text{ else } \ -\gamma$$

This final reward incorporates *p_0* (a quality threshold), *T_{max}* (maximum steps), and *\gamma* (bias for rewards/penalties), ensuring the agent seeks high-quality answers within reasonable limits.

Knowledge Transfer Prompt

A role-playing mechanism that helps LLMs effectively leverage previous answers without being constrained by them.

> "There is an answer to the question from another student: [previous answer]. You need to give a more satisfactory answer. DO NOT mention other students."

Key Innovation: Treats prior answer as "student's answer" to avoid override bias while encouraging improvement and knowledge synthesis.

Category	Methods	Param. ↓ Agent	Param. ↓ Infer.	s/sample ↓	Rouge-S ↑	BARTScore ↑	BLEURT ↑	BERTScore ↑	GPT-Rank↓
	Oracle (BARTScore)	–	–	–	0.33	-2.87	-0.38	73.23	-
	ChatGPT [Achiam *et al.*, 2023]	–	≈175B	–	0.39	-3.00	-0.26	76.23	-
LLMs	MOSS [Sun *et al.*, 2023]	–	16B	–	0.19	-3.69	-0.73	64.85	-
	Vicuna [Chiang *et al.*, 2023]	–	13B	–	0.27	-3.44	-0.61	69.60	-
	Alpaca [Taori *et al.*, 2023]	–	13B	–	0.29	-3.57	-0.53	71.46	-
	Baize [Xu *et al.*, 2023]	–	13B	–	0.20	-3.53	-0.66	65.57	-
	Open Assistant [LAION-AI, 2023]	–	12B	–	0.34	-3.45	-0.39	74.68	-
	Dolly2 [Conover *et al.*, 2023]	–	12B	–	0.16	-3.83	-0.87	62.26	-
	FLAN-T5 [Chung *et al.*, 2024]	–	11B	–	0.13	-4.57	-1.23	64.92	-
	Koala [Geng *et al.*, 2023]	–	7B	–	0.19	-3.85	-0.84	63.96	-
	Mosaic MPT [Team and others, 2023]	–	7B	–	0.14	-3.72	-0.82	63.21	-
	StableLM [Stability-AI, 2023]	–	7B	–	0.17	-4.12	-0.98	62.47	-
	ChatGLM [Du *et al.*, 2022]	–	6B	–	0.27	-3.52	-0.62	70.38	-
Ensemble	Classifier-OPT	125M	**13B**	**3.98**	0.27	-3.44	-0.61	69.60	2.84
	PairRanker [Jiang *et al.*, 2023]	400M	117B	44.41	0.32	-3.14	-0.38	73.03	2.25
	LLM-debate [Du *et al.*, 2023]	-	234B	56.58	0.27	-3.51	-0.55	71.59	3.16
	ReConcile [Chen *et al.*, 2024]	-	351B	55.75	0.24	-3.79	-0.71	68.61	3.68
	sampling-voting [junyou li *et al.*, 2024]	110M	234B	38.07	0.27	-3.39	-0.33	70.12	3.20
	DyLAN [Liu *et al.*, 2024]	-	>234B	115.68	0.28	-3.69	-0.64	70.90	3.85
	DER (Ours)	**125M**	17B	9.32	**0.35**(+9.38%)	**-3.14**(+0.00%)	**-0.31**(+6.06%)	**75.00**(+2.70%)	**2.02**

Experimental Results: Superior Performance

DER demonstrates significant improvements over state-of-the-art ensemble methods across multiple benchmarks:

85%

Compute reduction

Reduces computational overhead during inference from 117B to 17B compared to PairRanker while improving BERTScore by 2.7%.

16%

Accuracy Improvement

On mathematical reasoning (GSM8K), DER outperforms baselines with 77% fewer inference parameters.

98%

Performance

Achieves 98% of ChatGPT's performance with 10% as many inference parameters as ChatGPT

Key Findings & Ablation Studies

Rigorous analysis confirms each component contributes meaningfully to DER's superior performance:

Version	Param. ↓	BARTScore ↑	BLEURT ↑	BERTScore ↑
$T_{max} = 3$	**15B**	-3.15	-0.32	74.97
$T_{max} = 4$	17B	**-3.14**	**-0.31**	**75.00**
T_{max} (w/o Term.)	28B	-3.15	-0.34	73.32
$T_{max} = 5$	20B	-3.16	-0.32	74.94

Version	Agent	Infer.	BARTScore ↑	BLEURT ↑	BERTScore ↑
Two experts	125M	49B	-3.23	-0.33	74.80
One expert	125M	**17B**	**-3.14**	**-0.31**	**75.00**

Impact of Knowledge Transfer Prompt (KTP)

KTP improves BARTScore by 4.6% (-3.29 to -3.14), demonstrating effective knowledge transfer.

Effectiveness of MDP-based Routing

DER with KTP (BERTScore 75.00) outperforms random routing (BERTScore 72.88), showing a 2.12-point BERTScore improvement

Role of the Terminator

The terminator dynamically halts reasoning, reducing computational parameters by ~39% and boosting overall BERT score by an average of 1.68 points, balancing efficiency and output quality.

Effect of the number of experts selected

DER produces better generated answers by selecting one expert per state, with a 65% reduction in inference parameters

Conclusion: The Future of LLM Ensembling

DER represents a paradigm shift in how multiple LLM experts can be efficiently combined. By modeling ensemble reasoning as a Markov Decision Process with intelligent knowledge transfer, we achieve nearly 7-fold parameter reduction compared to existing methods while improving performance across diverse tasks.

Dynamic Adaptation

Routes are selected based on input questions, enabling flexible optimization for different task types and complexities.

Complementary Knowledge

Sequential knowledge transfer allows LLMs to build on each other's strengths, creating synergistic improvements.

Practical Efficiency

Achieves superior results with only 15% of inference parameters compared to ensemble methods using all LLM outputs.