BALANCING ACT: DIVERSITY AND CONSISTENCY IN LARGE LANGUAGE MODEL ENSEMBLES

Ahmed Abdulaal[1,2*] **Chen Jin**[1] **Nina Montaña-Brown**[1,†] **Aryo Pradipta Gema**[1,†]
Daniel Coelho de Castro[3] **Daniel C. Alexander**[2] **Philip Teare**[1] **Tom Diethe**[1]
Dino Oglic[1] **Amrutha Saseendran**[1]

[1]Centre for AI, Data Science & Artificial Intelligence, R&D, AstraZeneca, UK
[2]Centre for Medical Image Computing, UCL, London, UK
[3]Microsoft Research, Cambridge, UK

ABSTRACT

Ensembling strategies for Large Language Models (LLMs) have demonstrated significant potential in improving performance across various tasks by combining the strengths of individual models. However, identifying the most effective ensembling method remains an open challenge, as neither maximizing *output consistency* through self-consistency decoding nor enhancing *model diversity* via frameworks like 'Mixture of Agents' has proven universally optimal. Motivated by this, we propose a unified framework to examine the trade-offs between task performance, model diversity, and output consistency in ensembles. More specifically, we introduce a consistency score that defines a gating mechanism for mixtures of agents and an algorithm for mixture refinement to investigate these trade-offs at the semantic and model levels, respectively. We incorporate our insights into a novel inference-time LLM ensembling strategy called the Dynamic Mixture of Agents (DMoA) and demonstrate that it achieves a new state-of-the-art result in the challenging Big Bench Hard mixed evaluations benchmark. Our analysis reveals that cross-validation bias can enhance performance, contingent on the expertise of the constituent models. We further demonstrate that distinct reasoning tasks—such as arithmetic reasoning, commonsense reasoning, and instruction following—require different model capabilities, leading to inherent task-dependent trade-offs that DMoA can balance effectively.

1 INTRODUCTION

Frontier Large Language Models (LLMs) (Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023; OpenAI, 2023; Anthropic, 2024) continue to demonstrate ever-improving capabilities in natural language generation tasks, such as mathematical reasoning, creative writing, and interactive communication (Rafailov et al., 2024a). As foundation models, LLMs are trained on broad datasets at scale and are adaptable to a diverse range of downstream applications (Bommasani et al., 2021). However, further scaling model training is increasingly challenging in most practical settings due to prohibitive cost requirements (Stojkovic et al., 2024) and the necessity of retraining on several trillion tokens (Wang et al., 2024).

Ensembling LLM outputs has emerged as an increasingly promising alternative method for boosting performance in a variety of tasks including commonsense reasoning (Wang et al., 2022), instruction-following (Wang et al., 2024), and coding for mathematical reasoning (Project Numina, 2024). However, there is uncertainty around how LLM ensembles should be designed to achieve optimal performance across different tasks. A particularly salient issue seems to arise around the interplay between an ensemble's diversity and its consistency.

Recent work on Mixtures-of-Agents (MoAs) (Wang et al., 2024) has proposed that there exists an inherent 'collaborativeness' between LLMs, whereby an LLM can produce higher quality outputs when shown the outputs of other (heterogeneous) LLMs, irrespective of whether these other LLMs

[*] rmapabd@ucl.ac.uk

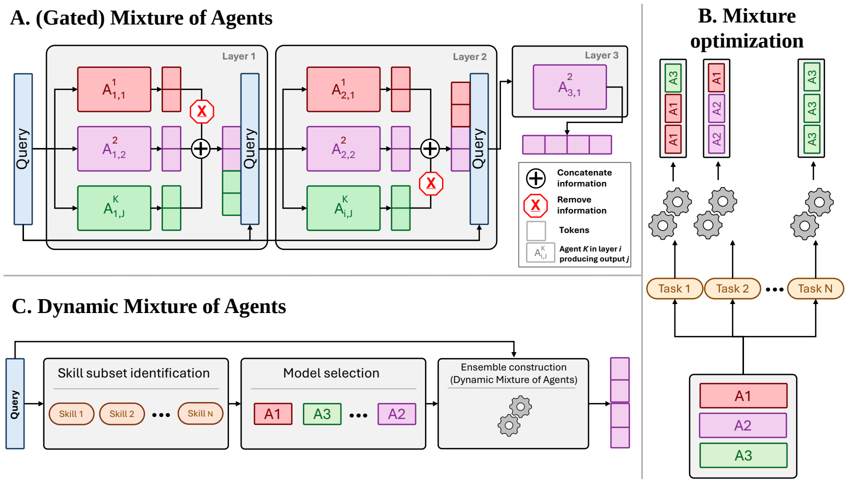
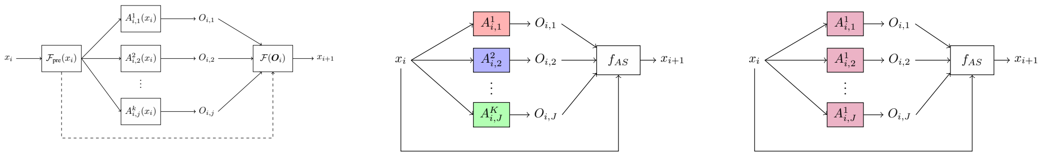


Figure 1: Our frameworks to evaluate the trade-offs between task performance and ensemble diversity/consistency. **A:** The **Gated Mixture of Agents experiment** introduces a divergence metric which we use to filter semantically inconsistent information from each layer. **B:** The **Mixture Optimization experiment** employs an algorithm (represented by the gear icon) to vary LLM mixtures to improve performance in a task-dependent manner; this allows us to investigate for possible trade-offs with other tasks. **C:** We propose an inference-time LLM ensembling strategy we call the **Dynamic Mixture of Agents (DMoA)** which identifies the skills required to respond to a query, and then selects a subset of models predicted to perform well given those skills. An inference-time ensemble is then constructed and executed. Comprehensive evaluations are conducted on benchmarks for instruction following, commonsense reasoning, and arithmetic reasoning to validate our methods.

are less capable than the main model. The authors demonstrated that using *maximally diverse* mixtures consistently led to higher Length-Controlled (LC) win rates on AlpacaEval 2.0 (Dubois et al., 2024). On the other hand, self-consistency-based (Wang et al., 2022) decoding strategies, whereby several outputs are sampled from a single LLM for a given query, continue to achieve state-of-the-art (SOTA) results in challenging tasks such as the AI Mathematics Olympiad (AIMO Prize, 2024; Project Numina, 2024). Indeed, it appears that *maximising consistency*, for example by combining self-consistency decoding with entailment verification, can further improve performance in a number of tasks (Sanyal et al., 2024a).

In this work, we systematically explore the trade-offs between task performance and ensemble diversity/consistency, and find strong empirical evidence that *neither* is universally optimal in general. We investigate the interaction between diversity and consistency at two levels: at the semantic level within fixed heterogeneous mixtures—drawing inspiration from hallucination-detection literature—and at the model level by optimizing the LLMs in a given mixture. We wish to systematize LLM ensembling strategies, then investigate these trade-offs at the semantic and model levels, and then opertaionalize our insights. To this end, our contributions are as follows: 1) *Unified perspective on LLM ensembling:* To enable clearer conceptualization and hypothesis generation for several ensembling strategies - we propose a unified framework that subsumes and systematizes current LLM ensembling approaches; 2) *Novel divergence metric ('Gated MoA'):* To investigate the effect of increasing output consistency in a given heterogeneous LLM mixture, we develop an 'EigenDivergence' metric, utilizing hallucination-detection-based scores in the sentence embedding space to enhance semantic consistency across LLM outputs. (Fig.1-A); 3) *Model-level ensemble optimization:* We formulate a simple optimization algorithm which allows us to systematically investigate the diversity/consistency trade-off at the level of mixture composition (Fig.1-B); 4) *Novel ensembling framework:* We propose a dynamic inference-time LLM ensembling strategy we call the 'Dynamic Mixture of Agents' (DMoA) (Fig.1-C) with which we achieve state-of-the-art performance in the Big Bench Hard mixed evaluations benchmark.



(a) Ensembling unit (b) MoA Framework (c) Self-Ensembling Framework

Figure 2: Illustration of a single layer ensembling unit and comparison of MoA and Self-Ensembling frameworks. (a) The ensembling unit shows the optional preprocessing function $\mathcal{F}_{\text{pre}}(\cdot)$ and aggregator function $\mathcal{F}(\boldsymbol{O}_i)$ that combines outputs $\{O_1, O_2, \ldots, O_j\}$ produced by agents $A^k_{i,j}$ into a single output. (b) The MoA framework uses different LLMs (represented by different colors) while (c) the Self-Ensembling framework uses a single LLM (represented by uniform color) to generate multiple outputs. Both use the aggregate and synthesize (f_{AS}) processing function to produce an output.

2 A UNIFIED PERSPECTIVE ON LLM ENSEMBLING

In the following, our use of the word 'layer' refers specifically to a *layer of LLMs* (Fig. 1-Panel A). In LLM ensembling, an input query x_i is used to sample a set of candidate outputs for each layer i of the framework from models $A^k_{i,j}$, where j corresponds to the output sentence index and k is the index for a given LLM. The input x_i may optionally be preprocessed by a function $\mathcal{F}_{\text{pre}}(\cdot)$, though most often this is the identity function and, as such, we henceforth refer to the input as x_i for notational simplicity. Outputs for layer i are then defined as $\boldsymbol{O}_i = \{O_{i,1}, O_{i,2}, \ldots, O_{i,j}\}$, where $O_{i,j} = A^k_{i,j}(x_i)$. A layer processing function is then applied to these outputs as $\mathcal{F}(\boldsymbol{O}_i)$. Layer processing functions can be described as layers in their own right, for instance as in Wang et al. (2024), and we adopt this nomenclature here. Processing functions $\mathcal{F}$ include (re-)rankers $f_R = \max_s S(\boldsymbol{O}_i)$, where $S(\cdot)$ is a scoring function assessing the quality of each response, majority voting $f_M = \arg\max_{O \in \boldsymbol{O}_i} \sum_{j=1}^{J} \mathbb{I}[O_{i,j} = O]$ (Wang et al., 2022), and verifiers $f_V = V(\boldsymbol{O}_i)$ (Chen et al., 2023b), where $V(\cdot)$ is a function directly predicting response quality. Processing functions are optionally parameterised by residual-like connections. For instance, in the 'aggregate and synthesise' (Wang et al., 2024) approach, the input query (x_1) is concatenated with a prompt to iteratively improve previous outputs as $f_{AS}(\boldsymbol{O}_i; x_1) = \odot(\boldsymbol{O}_i \oplus x_1)$, where the symbol $\oplus$ reflects concatenation of text and $\odot$ is an instruction to aggregate the previous answers and synthesise a novel response given the input query. Instantiations of this framework include self-consistency (Wang et al., 2022), which is a majority-voting, self-ensemble setup (which amounts to an argmax operation on the outputs of a layer)

$$x_{i+1} = f_M(\boldsymbol{O}_i) = \underset{O \in \boldsymbol{O}_i}{\arg\max} \sum_{j=1}^{J} \mathbb{I}[A^{k=1}_{i,j}(x_i) = O], \tag{1}$$

for a fixed index k (as one LLM is used to sample multiple outputs), and cascading-style ensembles such as FrugalGPT (Chen et al., 2023b)

$$x_{i+1} = f_V(O_{i,j=1}; x_1) = V(O_{i,j=1} \oplus x_1) \text{ s.t.} V(O_{i,j=1} \oplus x_1) \geq \tau, \tag{2}$$

whereby each layer contains a single LLM that attempts to produce an answer that is subsequently assessed for correctness by a verifier function. The process continues until the verifier attains a quality score above a pre-specified threshold τ (Chen et al., 2023b). More recently, Mixture of Agents (MoA) (Wang et al., 2024) was proposed as a multi-layer, multi-LLM ensemble which uses an aggregate and synthesise approach to produce a single output for each LLM at each layer:

$$x_{i+1} = f_{AS}(\boldsymbol{O}_i; x_1) = \odot \left(\oplus_{k=1}^{K} \left[A^k_{i,j=k}(x_i) \right] \oplus x_1 \right). \tag{3}$$

We are principally interested in multi-layer, multi-LLM ensembling with an aggregate and synthesise processing function (i.e. the MoA framework as in Eq. 3; Fig. 2b) and multi-layer self-ensembling with an aggregate and synthesise processing function (Fig. 2c):

$$x_{i+1} = f_{AS}(\boldsymbol{O}_i; x_1) = \odot \left(\oplus_{j=1}^{J} \left[A^{k=1}_{i,j}(x_i) \right] \oplus x_1 \right). \tag{4}$$

We believe this enables an appropriate comparison between the current state-of-the-art ensembling methods for instruction following benchmarks (Wang et al., 2024; Dubois et al., 2024; Zheng et al., 2024; Ye et al., 2023) and popular self-ensembling methods (Wang et al., 2022; Sanyal et al., 2024b). Nevertheless, a diverse LLM 'ensembling zoo' is enabled by the unified perspective proposed here. For instance, one could imagine a multi-layer self-ensembling framework with majority voting as the processing function, which would represent a multi-layer extension to the original self-consistency framework proposed by Wang et al. (2022), or a multi-layer, multi-LLM ensemble with a verifier function where the LLMs are increasingly more capable in each successive layer (which would represent a multi-LLM extention to work by Chen et al. (2023b)). Whether every prompt completion should be cross-validated to improve response quality, and with which LLM ensembling architecture, remains an open question.

3 METHODS

Our overarching hypothesis is that task-dependent trade-offs exist between performance and ensemble diversity/consistency, informing our experimental progression. First, in the **Gated Mixture of Agents (GMoA) experiment**, we assess the diversity-consistency trade-off within a fixed mixture by introducing a divergence metric based on the EigenScore for hallucination detection (Chen et al., 2024a) to *enforce greater semantic consistency among LLM outputs*. Second, in the **Mixture Optimization experiment**, we examine this trade-off at the level of mixture composition, proposing an algorithm to *systematically vary the included LLMs to explore how different combinations impact task performance*. Results from the first two experiments provide insights which guide the design of our third approach: an inference-time ensembling strategy, the **Dynamic Mixture of Agents (DMoA)**, which constructs task-dependent ensembles based on the specific skills required to produce high-quality outputs.

3.1 GATED MIXTURE OF AGENTS VIA DIVERGENCE FILTERING

In this section we describe a novel divergence metric which we leverage in Sec. 4.1 to enforce greater semantic consistency by filtering out some LLM outputs (Fig.1-Panel A). Motivated by the observation that sampling multiple outputs for a fixed input is useful for estimating sequence-level uncertainty (Manakul et al., 2023), Chen et al. (2024a) proposed an EigenScore to detect LLM hallucinations based on the consistency of semantic information in the outputs. For K sampled outputs, the covariance of the sentence embeddings can be calculated as

$$\mathbf{\Sigma} = \boldsymbol{Z}^\top \cdot \boldsymbol{I}_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^\top \cdot \boldsymbol{Z}, \tag{5}$$

where $\boldsymbol{Z} \in \mathbb{R}^{d\times K}$ is a matrix of d-dimensional sentence embeddings, $\mathbf{1}_d$ is an all-one column vector, $\boldsymbol{I}_d$ is a d-dimensional identity matrix, and $\mathbf{\Sigma} \in \mathbb{R}^{K\times K}$ is a covariance matrix capturing semantic consistency in sentence-embedding space. The EigenScore is then the logarithmic determinant of $\mathbf{\Sigma}$ or, equivalently, the average logarithm of the eigenvalues

$$E(\mathbf{\Sigma}|\boldsymbol{x};\boldsymbol{\theta}) := \frac{1}{K}\log\det(\mathbf{\Sigma} + \alpha\cdot\boldsymbol{I}_K) = \frac{1}{K}\sum_i^K \log(\lambda_i), \tag{6}$$

where $\boldsymbol{x}$ is the input context, $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_K\}$ are the parameters of the LLMs in an MoA (which may be shared in the case of a single LLM producing multiple outputs), and $\alpha \cdot \boldsymbol{I}_K$ is a regularization term to ensure the covariance matrix is full rank. Eigenscores were demonstrated to be more robust than perplexity (Ren et al., 2022), entropy (Kadavath et al., 2022), length-controlled entropy (Malinin & Gales, 2020; Lin et al., 2023), and lexical similarity scores (Lin et al., 2022) for consistency-based hallucination detection and are therefore operationalized here as a semantic consistency metric (Chen et al., 2024a). It should be noted that the logarithm determinant operation in Eq. 6 has a strong relationship to information entropy (Cover, 1999). Indeed, for a multivariate Gaussian $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, the Shannon differential entropy is defined as

$$h(\mathbf{X}) = \frac{1}{2}\log\det\mathbf{\Sigma} + \frac{d}{2}(\log 2\pi + 1) = \frac{1}{2}\sum_i^d \log\lambda_i + C, \tag{7}$$

where d is the dimension of the variable $\mathbf{X}$ and C is a constant term (Zhouyin & Liu, 2021). The EigenScore can therefore be interpreted as the differential entropy in sentence embedding space. Chen et al. (2024a) used the final token embeddings in the middle layer as a proxy for sentence embeddings, however this limited analyses to a single open-source LLM, as this required access to the internal model states. To enable evaluations between multiple (open- and closed-source) LLMs, we project LLM outputs into a sentence-embedding space as $\boldsymbol{Z} = e(\boldsymbol{S})$, where $\boldsymbol{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_K\}$ is a set of K output sequences and $e(\cdot)$ is an embedding function. We demonstrate strong concordance between 'internal' and 'external' eignscores in Appendix F. We additionally introduce EigenDivergence as a measure of an individual output's contribution to overall semantic consistency within an MoA. For a given MoA with K outputs, we calculate the EigenDivergence of the i-th output as the difference between the overall EigenScore and the EigenScore computed without that output:

$$D_i(\boldsymbol{\Sigma}|\boldsymbol{x};\boldsymbol{\theta}) = E(\boldsymbol{\Sigma}|\boldsymbol{x};\boldsymbol{\theta}) - E_{-i}(\boldsymbol{\Sigma}{-}i|\boldsymbol{x};\boldsymbol{\theta}{-}i), \tag{8}$$

where $E(\boldsymbol{\Sigma}|\boldsymbol{x};\boldsymbol{\theta})$ is the EigenScore calculated using all K outputs, and $E_{-i}(\boldsymbol{\Sigma}{-}i|\boldsymbol{x};\boldsymbol{\theta}{-}i)$ is the EigenScore calculated using $K-1$ outputs, excluding the i-th output. The EigenDivergence thus quantifies the extent to which each individual output contributes to or detracts from the overall semantic consistency of the MoA. As per Eq. 7, the EigenDivergence can be interpreted as the change in differential entropy when removing an output from the ensemble:

$$D_i(\boldsymbol{\Sigma}|\boldsymbol{x};\boldsymbol{\theta}) \propto h(\boldsymbol{\Sigma}) - h(\boldsymbol{\Sigma}_{-i}). \tag{9}$$

This formulation provides an information-theoretic interpretation of EigenDivergence: it quantifies the amount of unique information contributed by each output to the overall ensemble, and is therefore closely related to information gain (MacKay, 2003). A further discussion of this relationship is given in Appendix G. A large positive EigenDivergence indicates that removing the i-th output significantly reduces the entropy (or uncertainty) of the ensemble, suggesting that this output contributes novel or diverse information. Conversely, a negative EigenDivergence suggests that the output is more closely aligned with the information already present in the ensemble.

3.2 Mixture Optimization

To investigate the tradeoff between diversity and consistency at the level of mixture composition, we require an optimization approach with three desiderata: 1) The algorithm is cost efficient; 2) It optimizes for stronger performance for a chosen downstream task; 3) It allows for interpolation between multi-LLM and self-ensembling mixtures. To this end, we propose a simple algorithm to optimize a mixture composition towards higher performance for a given benchmark as follows: For a GMoA (Fig. 1-Panel A), let P_i be the performance metric for run i. The performance difference across runs is then

$$\Delta P = P_{i+1} - P_i. \tag{10}$$

Let $U_i(m)$ be the usage of model m within the mixture for run i, then the usage difference across runs is

$$\Delta U(m) = U_{i+1}(m) - U_i(m). \tag{11}$$

The delta for each model (ΔM) is then its proportional impact on the performance change

$$\Delta M = \left(\frac{\Delta U(m)}{\sum_m |\Delta U(m)|}\right) \times \Delta P \tag{12}$$

where Eq. 12 assumes that each model's impact is directly proportional to its usage change relative to the total usage change. A high model delta indicates that either increasing model usage correlated with an increase in performance, or decreasing its usage correlated with a decrease in performance. A negative model delta indicates that increased model usage correlated with a decrease in performance, or decreasing its usage correlated with an increase in performance. In each run, the model with the lowest delta is replaced by a 'clone' of the model with the highest delta, thereby adapting the mixture towards models that demonstrate the strongest positive impact on overall performance. Clones sample a different output for the same input. A full description of the algorithm is given in Appendix C.

Table 1: Performance impact of EigenDivergence filtering across benchmarks. All ensemble experiments are run three times and average scores are reported with standard deviations. † denotes our replication of results.

	Instruction Following		**Arithmetic Reasoning**		**Commonsense Reasoning**		
Model	**AlpacaEval 2.0**	**MT-Bench**	**GSM8K**	**MATH**	**CSQA**	**ARC-C**	**ARC-E**
GMoA (Ours)	58.66 ± 0.29	8.97 ± 0.18	**94.23** ± 0.29	**56.35** ± 0.33	**85.20** ± 0.22	**92.32** ± 0.38	93.75 ± 0.14
MoA	**59.50** ± 0.08	**9.19** ± 0.07	93.87 ± 0.35	55.22 ± 0.29	84.32 ± 0.31	91.85 ± 0.14	**94.31** ± 0.32
Llama-3-70B	34.4	8.8	93.0	50.4	83.8	90.5†	94.1
Qwen-1.5-110B	43.9	8.9	85.4	49.6	82.1†	69.6	93.9†
Qwen-1.5-72B	36.6	8.4	79.5	34.1	83.2†	65.9	92.7†
WizardLM-8x22B	51.3	8.8	81.6	22.7	69.0†	62.5	90.1†
Mixtral 8x22B	30.9	8.8	83.7	41.7	81.7†	70.7	91.8†
DBRX-Instruct	25.4	8.4	72.8	32.5	82.2†	68.9	89.7†
GPT-4 Omni (05/13)	57.5	9.19	94.1†	61.2%†	88.6†	94.6†	94.3†

3.3 Dynamic Mixtures of Agents (DMoAs)

Building on the insights from GMoA and Mixture Optimization on balancing semantic consistency and diversity for task-specific performance, we propose the Dynamic Mixture of Agents (DMoA) to create adaptive, skill-targeted ensembles at inference time. The preceding experiments demonstrate that balancing diversity and consistency is crucial yet varies by task type: Given a set of queries $\mathcal{Q} = q_i$ and models $\mathcal{M} = m_i$, for each query $q_j \in \mathcal{Q}$, we first identify the required skills $\mathcal{S} = f_s(q_j; \boldsymbol{\theta})$ (based on Sec. 4.4; Insight 3), where $\boldsymbol{\theta}$ are optionally learnable parameters. Next, we select a subset of models $\mathcal{MS} = f_m(\mathcal{S}; q_j, \boldsymbol{\theta}) \subseteq \mathcal{M}$ predicted to perform well given these skills (based on Sec. 4.4; Insight 2). This forms a preprocessing function for query routing, $\mathcal{F}\text{pre}(q_j) = f_m \circ f_s$ (as introduced in Sec. 2). Each model in $\mathcal{M}_{\mathcal{S}}$ generates an output, which we combine using the aggregation and synthesis method (Eq. 3), leveraging cross-validation bias to improve results (based on Sec. 4.4; Insight 1). Unlike traditional LLM routing which often selects a single optimal model for a query (Lu et al., 2023; Shnitzer et al., 2023; Liu et al., 2023), the DMoA constructs an ensemble of models queried at inference time. We assess the DMoA framework on Big Bench Hard (BBH) (Suzgun et al., 2022), a set of 23 challenging tasks including language understanding, algorithmic reasoning, and multistep arithmetic. Details of the experimental setup are provided in Appendix E.

4 Experiments

4.1 Gated Mixture of Agents via Divergence Filtering

Experimental setup We construct an MoA (Fig. 2b) using the same open-source models proposed by Wang et al. (2024). Namely, we use Llama-3-70B-Instruct (Touvron et al., 2023), Qwen1.5-72B-Chat (Bai et al., 2023), Qwen1.5-110B-Chat (Bai et al., 2023), Mixtral-8x22B-v0.1 (Jiang et al., 2024), WizardLM-8x22B (Xu et al., 2023), and dbrx-instruct (The Mosaic Research Team, 2024). We construct the 'MoA-Lite' variant of this meta-architecture, which contains 2 MoA layers and uses Qwen1.5-72B-Chat as the 'aggregate and synthesize' processing function (Eq. 3). To investigate the interplay between semantic consistency and diversity and its effect on task performance, we use EigenDivergence to filter the two least semantically consistent outputs from a given MoA layer. The idea is to establish a balance between diversity of outputs and consistency of semantic information across the ensemble. By removing the outputs with the highest EigenDivergence, we aim to maintain a core set of semantically consistent responses while still preserving some level of diversity from the heterogeneous LLMs. We then compare the performance of standard MoAs against their divergence-filtered variants, which we call Gated Mixtures of Agents (GMoAs). Additional implementation details can be found in Appendix B, and a description of the relationship between GMoAs and classical Mixtures of Expert (MoE) approaches (Shazeer et al., 2017) is given in Appendix B.5.

Results Results of the EigenDivergence filtering experiment are shown in Table 1. Whilst the LLM ensembles broadly outperform their individual constituent LLMs across all tasks, EigenDivergence filtering demonstrates different effects across different types of reasoning tasks. **Instruction following:** AlpacaEval 2.0 is a popular length-controlled, reference-free instruction following benchmark (Dubois et al., 2024). Here, the GMoA (filtered) model performs substantially worse with a 0.84% absolute degradation in performance, with a similar decline of 0.22% noted on MT-Bench (Zheng et al., 2024). **Arithmetic reasoning:** Increasing semantic consistency by performing filtering does not degrade performance for arithmetic reasoning, with the GMoA demonstrating a 0.36% absolute *improvement* in performance for the GSM8K benchmark (Cobbe et al., 2021), with a 1.12% performance gain in the MATH benchmark (Hendrycks et al., 2021). **Commonsense reasoning:** In the commonsense reasoning tasks, the impact of EigenDivergence filtering is nuanced. For example, the GMoA model shows an absolute improvement of 0.88% on the CSQA benchmark (Talmor et al., 2018), indicating that filtering enhances the model's ability to manage tasks requiring broad knowledge and logical inference. However, this effect is less consistent across other benchmarks. On ARC-E (Clark et al., 2018), where most models already exhibit high accuracy, filtering might inadvertently remove correct but diverse responses, leading to a slight underperformance compared to the unfiltered MoA model. This suggests that in tasks where models have already reached a high level of proficiency, the filtering process could sometimes hinder rather than help by excluding potentially valuable outputs. Conversely, in more challenging tasks like ARC-C (Clark et al., 2018), where inconsistency is more likely to correlate with errors, filtering proves beneficial. The differential impact of EigenDivergence filtering across benchmarks appears to support our initial hypothesis of a task-dependent trade-off between diversity and consistency. This is noted at the level of semantic consistency within a fixed (heterogeneous) LLM ensemble.

4.2 Mixture Optimization

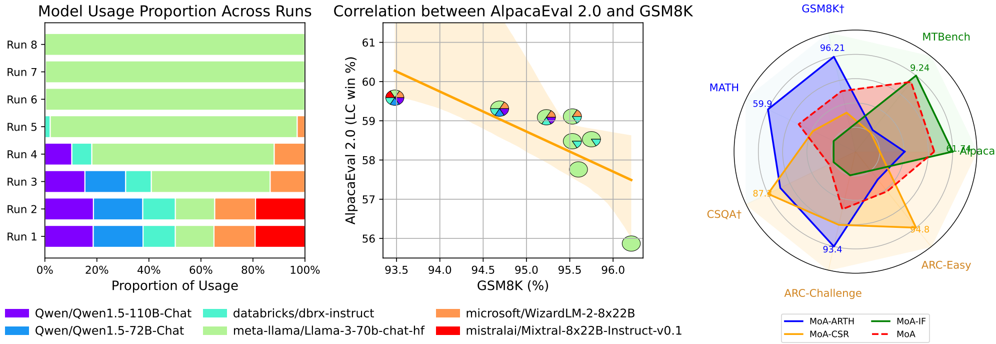


Figure 3: **Left:** Model usage and performance comparison. The algorithm was used to optimize for arithmetic reasoning on the GSM8K benchmark. The left-most figure shows usage proportion across runs. In this instance Llama-3-70b-chat demonstrated consistently positive model deltas across runs. The central-plot illustrates that as mixture performance increased for GSM8K, length-controlled winrates for AlpacaEval 2.0 appreciably decreased. **Right:** Performance trade-offs in LLM ensemble optimization across reasoning tasks. MoA-ARTH = Arithmetic reasoning optimized mixture, MoA-CSR = Commonsense reasoning optimized mixture, and MoA-IF = Instruction following optimized mixture. † Represents benchmarks used to optimize an ensemble.

Experimental setup For each task, we start with the same LLM composition and MoA architecture as in Sec. 4.1 and run the optimization algorithm for each of instruction following, arithmetic reasoning, and commonsense reasoning. For arithmetic reasoning tasks, we optimize the ensemble performance on the GSM8K benchmark. For commonsense reasoning, we optimize performance on CSQA, and for instruction following we optimize on AplacaEval 2.0. We use the same sampling scheme as in Sec. 4.1 (described in Appendix B.4) to generate LLM outputs, and compare the relative performance of optimized mixtures to the baseline heterogeneous MoA.

Results Optimizing a mixture for a given benchmark demonstrates a clear trade-off with other reasoning tasks. Fig. 3-left/center illustrates an example run of the optimization algorithm on the GSM8K benchmark and the effect of adapting the mixture for improved arithmetic reasoning on instruction following performance as measured by AlpacaEval 2.0. As the mixture becomes less diverse, arithmetic reasoning improves. Additionally, there is a negative correlation between performance on GSM8K and AlpacaEval 2.0; The baseline MoA scores 93.48% on GSM8K and 59.59% on AlpacaEval 2.0, whereas the optimized mixture is a self-ensemble which scores 96.21% on GSM8K and 55.87% on AlpacaEval 2.0. Fig. 3-right illustrates the performance trade-offs more broadly; the "MoA-IF" configuration, optimized for instruction following, yielded the highest AlpacaEval score of 61.74, but its arithmetic reasoning performance dropped significantly, with a GSM8K score of 91.36. In commonsense reasoning tasks, the "MoA-CSR" configuration, optimized for these tasks, performed best on the CSQA benchmark with a score of 87.3. However, this focus led to decreased performance in both arithmetic reasoning and instruction following, with GSM8K and AlpacaEval scores of 92.4 and 52.83, respectively. These findings demonstrate that it is challenging to achieve a universally optimal LLM ensemble, as improvements in one area often necessitate compromises in others.

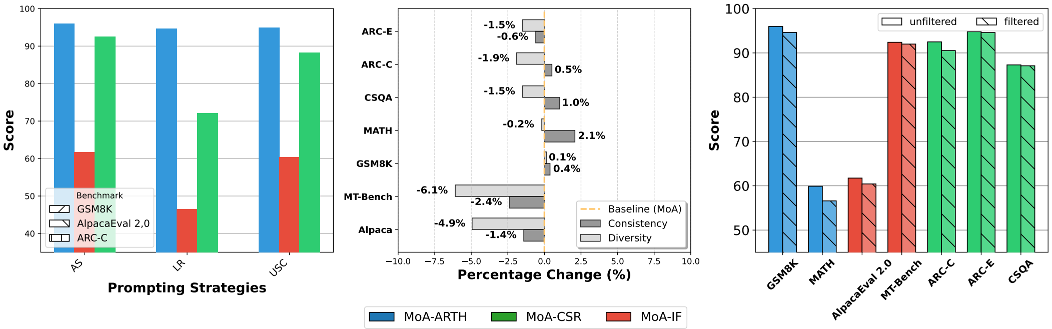


Figure 4: **Left:** The aggregation and synthesis (AS) strategy (which explicitly performs a critical appraisal of outputs from the previous layer in the ensemble) outperforms both simple LLM-based ranking (LR), and universal self-consistency (USC), which selects for the most consistent information. **Center:** Enforcing higher semantic diversity by removing more consistent outputs degrades performance across all reasoning tasks relative to both the baseline ensemble and the lower semantic diversity ensemble. **Right:** Filtering already specialized ensembles degrades performance across all reasoning tasks, and suggests that specialized knowledge retained in unfiltered ensembles contributes meaningfully to performance.

4.3 Ablation studies

We conduct a set of additional experiments which further characterize the effect of semantic diversity in LLM mixtures, the effect of filtering in specialised ensembles, and different processing functions for different reasoning tasks.

Aggregation and synthesis outperform LLM-based ranking and self-consistency. We compared the aggregation and synthesis (AS) function $f_{AS}(\cdot)$ with LLM-based ranking $f_R(\cdot)$ and self-consistency $f_M(\cdot)$ as described in Sec. 2. Specifically, we used the recently proposed 'universal' self-consistency (USC) by Chen et al. (2024b), which extends self-consistency to open-ended language generation tasks (experimental setup in Appendix D). Our results show that AS outperforms both ranking and USC, with ranking particularly detrimental for commonsense reasoning and instruction following (Fig. 4-left). This suggests that combining multiple results through critical appraisal is preferable to performing an argmax operation or selecting repeating information in open-ended tasks (Chen et al., 2024b). **Higher semantic diversity leads to worse performance across reasoning tasks.** We compared the baseline MoA (Sec. 4.1) with two GMoA variants: one with the two most semantically divergent outputs removed (maximizing consistency), and one with the two most consistent outputs removed (maximizing diversity). We found that higher semantic diversity in a heterogeneous ensemble degrades performance across all tasks (Fig. 4-center), indicating that some semantic consistency is necessary for high-quality results, even in open-ended instruction-following queries. **Filtering specialized ensembles degrades performance.** Applying semantic filtering to ensembles already optimized for a specific task leads to slight performance decreases across arithmetic reasoning (MoA-ARTH), instruction following (MoA-IF), and commonsense reasoning (MoA-CSR), with reductions ranging from 0.04% to 3.3% (Fig. 4-right). This suggests that the specialized knowledge retained in unfiltered, optimized ensembles contributes meaningfully to task performance and should therefore be maintained.

4.4 Dynamic Mixtures of Agents (DMoAs)

In this section, we operationalize the following insights from our experiments: **Insight 1 - Inductive cross-validation bias: Reasoning traces validated by multiple hypotheses are more likely to be correct.** As shown in Sec. 4.1, LLM ensembles outperform individual models regardless of divergence filtering, across both open- and close-ended tasks. Maximizing semantic diversity harms performance (Sec. 4.3), indicating that some cross-validation between outputs is necessary for high-quality results, even in open-ended instruction-following tasks (Wang et al., 2022; 2024). **Insight 2 - Task-dependent LLM expertise is crucial:** Contrary to prior work (Wang et al., 2022), we found in Sec. 4.1 that *removing* information from an ensemble can improve task-specific performance. Sec. 4.2 shows that optimizing a mixture for a specific task enhances performance on similar tasks but degrades it on dissimilar ones, especially between instruction following and arithmetic or commonsense reasoning. Our ablations (Sec. 4.3) demonstrate that removing information from specialized ensembles diminishes their target performance. **Insight 3 - Task-specific skills reside in different subspaces:** Mixtures which are optimized for one type of reasoning task tend to underperform in others (Sec. 4.2). Additionally, in our third ablation (Sec. 4.3), we showed that while aggregation and synthesis outperform other processing functions, tasks differ in sensitivity to the processing method used.

Table 2: Performance on the Big Bench Hard (BBH) benchmark, evaluating accuracy, normalized accuracy, and best/worst subset results. The LLM ensembles use Qwen2-72B-Instruct as the final aggregator, with the exception of DMoA/Sonnet, which uses Claude 3.5 Sonnet.

	Acc. (%)(↑)	**Norm. acc.** (%)(↑)	**Best subset**	**Worst subset**
DMoA/Sonnet (ours)	**93.12%**	**91.85%**	temporal sequences	causal judgement
Claude 3.5 sonnet	93.11%	90.20%	tracking 5 objs	causal judgement
DMoA (ours)	**86.05%**	**83.63%**	hyperbaton	causal judgement
MoA	83.81%	79.25%	web of lies	dyck languages
meta-llama/Llama-3-70b-chat-hf	84.55%	78.28%	temporal sequences	formal fallacies
microsoft/WizardLM-2-8x22B	82.12%	75.85%	web of lies	causal judgement
mistralai/Mixtral-8x22B-Instruct-v0.1	81.69%	75.11%	hyperbaton	causal judgement
Qwen/Qwen1.5-110B-Chat	76.40%	68.45%	web of lies	formal fallacies
Qwen/Qwen1.5-72B-Chat	75.24%	66.94%	web of lies	formal fallacies
databricks/dbrx-instruct	69.84%	59.29%	movie recommendation	sports understanding

Results Table 2 shows that the MoA framework outperforms individual models, and the DMoA surpasses the MoA, achieving 83.63% normalized accuracy—92.7% of the performance of Claude 3.5 Sonnet, the current state-of-the-art (SoTA) model for BBH (Anthropic, 2024). When Claude 3.5 Sonnet is used as the final aggregator in the DMoA, it achieves a new SoTA normalized accuracy of 91.85%. Despite differences in architectures, data mixtures, and training regimes, many models perform poorly on tasks requiring causal judgment or evaluating formal fallacies, performing better on simpler sequence-based tasks. This limitation persists even in ensembles, suggesting that novel expertise cannot be acquired through simple ensembling; if the underlying models lack the necessary skills to solve a task, their ensemble cannot compensate for that deficiency. The DMoA balances the trade-offs identified in prior experiments.

5 Related Work

LLM Reasoning Chain of Thought (CoT) reasoning was introduced as a few-shot 'multi-step' prompting technique encouraging LLMs to produce intermediate reasoning traces to improve performance on reasoning tasks (Wei et al., 2022), with both few-shot (Rubin et al., 2021) and zero-shot (Kojima et al., 2022) variants. Decomposed prompting (Khot et al., 2022) tackles complex tasks by breaking them into simpler sub-tasks, each delegated to dedicated LLMs. Similarly, least-to-most prompting (Zhou et al., 2022) decomposes problems into sub-tasks, but each solution depends on the answers to previous sub-tasks. Other approaches integrate multiple reasoning traces (Fu et al., 2022; Wang et al., 2022; Li et al., 2022) or select traces based on similarity (Rubin et al., 2021; Lu et al., 2022), diversity (Zhang et al., 2022), or entailment (Sanyal et al., 2024a). The Tree of Thought (ToT) framework (Yao et al., 2024) generalizes CoT by having LLMs self-evaluate multiple reasoning paths, while Graph of Thought (GoT) frames reasoning as a graphical task by producing a

’thought graph’ (Yao et al., 2023). However, most of these frameworks restrict analysis to a single LLM for each task.

LLM Ensembling Jiang et al. (2023) performed pairwise comparisons between model outputs to select the higher quality response. A number of works trained models which predicted the best performing LLM from a fixed set for a given user input (Lu et al., 2023; Shnitzer et al., 2023; Wang et al., 2023a). Other studies have investigated combining outputs from different models by a synthesis of results (Jiang et al., 2023), or by averaging model output probability distributions (Huang et al., 2024). A related line of work proposed symmetric (Du et al., 2023) or asymmetric (Liang et al., 2023; Chan et al., 2023; Chen et al., 2023a) frameworks to improve reasoning through problems. Wang et al. (2024) proposed the Mixture-of-Agents (MoA) approach - a multilayer framework with different LLM agents within each layer. Each agent iteratively improved the outputs from the previous layer to achieve higher quality outputs. This represented the assumed benefit of model heterogeneity on downstream tasks.

Hallucination Detection Models which produce inconsistent results (by sampling multiple outputs given a fixed query) have been thought to be ‘indecisive’ about how to respond to the query and more likely to produce hallucinations (Kuhn et al., 2023; Manakul et al., 2023; Raj et al., 2023; Chen et al., 2024a). Kersting et al. (2024) perturbed LLM input queries in a non-semantically meaningful way and sampled a single output for each perturbation, following which a consistency metric was computed to detect hallucinations. Whilst consistency metrics have included (normalized)entropy (Kadavath et al., 2022; Malinin & Gales, 2020; Lin et al., 2023), perplexity (Ren et al., 2022), and lexical similarity scores (Lin et al., 2022), several works have leveraged LLMs to self-evaluate their responses directly (Manakul et al., 2023; Kadavath et al., 2022; Cohen et al., 2023). Jesson et al. (2024) framed hallucination detection as a Bayesian task by defining hallucinations as low-probability predictions under a true latent parameter. Broadly, hallucination detection work does not explicitly perform hallucination mitigation (Kuhn et al., 2023; Chen et al., 2024a; Jesson et al., 2024). For instance, whilst the INSIDE score proposed by Chen et al. (2024a) was used to detect the likelihood of a hallucination, the score itself was not used for hallucination mitigation.

6 Discussion, Limitations, and Conclusion

In this work, we systematically explored the interplay between diversity and consistency in LLM ensembles and proposed a unified framework that subsumes existing ensembling approaches. Through the development of the EigenDivergence metric, we assessed the impact of semantic consistency within ensembles and found that while strong consistency improves performance in tasks like arithmetic and commonsense reasoning, it can impair instruction-following proficiency. Conversely, we observed that introducing an optimal degree of diversity can partially mitigate these limitations, enhancing instruction-following capabilities at the expense of reasoning and mathematical performance. Our mixture optimization experiments revealed inherent trade-offs when tailoring ensembles to specific tasks, emphasizing the challenge of balancing diversity and consistency. Our proposed Dynamic Mixture of Agents (DMoA) achieves state-of-the-art performance on the challenging Big Bench Hard (BBH) benchmark by dynamically selecting models based on task-specific skills. The ability to scale inference-time compute based on query requirements opens new directions for optimizing LLM performance across a diverse range of tasks, and is a natural avenue of future research.

The trade-offs in performance across benchmarks can be explained by Goodhart’s law, which states that statistical regularities collapse under pressure for control (Goodhart, 1984). For instance, the OpenAI o1-preview model dramatically outperformed gpt-4o at arithmetic reasoning, yet underperformed it in ‘personal writing’ (OpenAI, 2024). Additionally, overoptimizing proxy reward models in RLHF settings has led to decreased utility relative to ground-truth rewards (Gao et al., 2023), a phenomenon also observed with direct alignment algorithms like DPO (Rafailov et al., 2024a;b). Our findings suggest this overoptimization extends to ensembles of pretrained LLMs. Specifically, Extremal Goodhart’s law (Manheim & Garrabrant, 2018) indicates that optimizing for extreme proxy values—such as human-preference-based instruction following—can degrade performance in unrelated tasks like arithmetic reasoning. This may result from model insufficiency, where the proxy metric oversimplifies the target skill (Zheng et al., 2024; Koo et al., 2023; Wu & Aji, 2023; Wang et al., 2023b), or because optimization pushes models into regions where underlying processes differ between skills. Consequently, models highly optimized for a specific task such as instruction following may unexpectedly underperform in other reasoning tasks.

References

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. Cognitive science, 9(1):147–169, 1985.

AIMO Prize. Aimo prize, 2024. URL https://aimoprize.com/. Accessed: 2024-07-12.

Anthropic. Introducing claude 3.5 sonnet, 2024. URL https://www.anthropic.com/news/claude-3-5-sonnet. Accessed: 2024-07-12.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258, 2021.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better llm-based evaluators through multi-agent debate. arXiv preprint arXiv:2308.07201, 2023.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. Inside: Llms' internal states retain the power of hallucination detection. arXiv preprint arXiv:2402.03744, 2024a.

Justin Chih-Yao Chen, Swarnadeep Saha, and Mohit Bansal. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. arXiv preprint arXiv:2309.13007, 2023a.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. arXiv preprint arXiv:2305.05176, 2023b.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language models. In ICML 2024 Workshop on In-Context Learning, 2024b.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240):1–113, 2023.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. Lm vs lm: Detecting factual errors via cross examination. arXiv preprint arXiv:2305.13281, 2023.

Thomas M Cover. Elements of information theory. John Wiley & Sons, 1999.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. arXiv preprint arXiv:2305.14325, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. arXiv preprint arXiv:2404.04475, 2024.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. arXiv preprint arXiv:1805.04833, 2018.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. arXiv preprint arXiv:1707.02633, 2017.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In The Eleventh International Conference on Learning Representations, 2022.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In International Conference on Machine Learning, pp. 10835–10866. PMLR, 2023.

Charles AE Goodhart. Problems of monetary management: the UK experience. Springer, 1984.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. arXiv preprint arXiv:2103.03874, 2021.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. arXiv preprint arXiv:1805.06087, 2018.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. arXiv preprint arXiv:1904.09751, 2019.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Bing Qin, and Ting Liu. Enabling ensemble learning for heterogeneous large language models with deep parallel collaboration. arXiv preprint arXiv:2404.12715, 2024.

Hugging Face. Open llm leaderboard: About. https://huggingface.co/docs/leaderboards/open_llm_leaderboard/about, 2023. Accessed: 2024-09-16.

Andrew Jesson, Nicolas Beltran-Velez, Quentin Chu, Sweta Karlekar, Jannik Kossen, Yarin Gal, John P Cunningham, and David Blei. Estimating the hallucination rate of generative ai. arXiv e-prints, pp. arXiv–2406, 2024.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. arXiv preprint arXiv:2401.04088, 2024.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. arXiv preprint arXiv:2306.02561, 2023.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022.

Nicholas S Kersting, Mohammad Rahman, Suchismitha Vedala, and Yang Wang. Harmonic llms are trustworthy. arXiv preprint arXiv:2404.19708, 2024.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. arXiv preprint arXiv:2210.02406, 2022.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. Advances in neural information processing systems, 35:22199–22213, 2022.

Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. Benchmarking cognitive biases in large language models as evaluators. arXiv preprint arXiv:2309.17012, 2023.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. arXiv preprint arXiv:2302.09664, 2023.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making large language models better reasoners with step-aware verifier. arXiv preprint arXiv:2206.02336, 2022.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent debate. arXiv preprint arXiv:2305.19118, 2023.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models. arXiv preprint arXiv:2305.19187, 2023.

Zi Lin, Jeremiah Zhe Liu, and Jingbo Shang. Towards collaborative neural-symbolic graph semantic parsing via uncertainty. Findings of the Association for Computational Linguistics: ACL 2022, 2022.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Making ppo even better: Value-guided monte-carlo tree search decoding. arXiv preprint arXiv:2309.15028, 2023.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. arXiv preprint arXiv:2311.08692, 2023.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. arXiv preprint arXiv:2209.14610, 2022.

David JC MacKay. Information theory, inference and learning algorithms. Cambridge university press, 2003.

Andrey Malinin and Mark Gales. Uncertainty estimation in autoregressive structured prediction. arXiv preprint arXiv:2002.07650, 2020.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. arXiv preprint arXiv:2303.08896, 2023.

David Manheim and Scott Garrabrant. Categorizing variants of goodhart's law. arXiv preprint arXiv:1803.04585, 2018.

Open LLM Leaderboard Team. Open llm leaderboard. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024. Accessed: 2024-09-16.

OpenAI. Gpt-4 technical report, 2023.

OpenAI. New embedding models and api updates, January 2024. URL https://openai.com/index/new-embedding-models-and-api-updates/. Accessed: 2024-06-27.

OpenAI. Language model embeddings. https://platform.openai.com/docs/guides/embeddings/what-are-embeddings, 2024a. Accessed: 2024-08-15.

OpenAI. Gpt-3.5 turbo. https://platform.openai.com/docs/models/gpt-3-5-turbo, 2024b. Accessed: 2024-09-16.

OpenAI. Hello gpt-4o. https://openai.com/index/hello-gpt-4o/, 2024c. Accessed: 2024-09-16.

OpenAI. Learning to reason with large language models. https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2024-09-18.

Project Numina. Publications, 2024. URL https://projectnumina.ai/publications/. Accessed: 2024-07-12.

Qwen Team. Qwen1.5-110b: The first 100b+ model of the qwen1.5 series. https://qwenlm.github.io/blog/qwen1.5-110b/, 2024. Accessed: 2024-09-16.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sikchi, Joey Hejna, Bradley Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. arXiv preprint arXiv:2406.02900, 2024a.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems, 36, 2024b.

Harsh Raj, Vipul Gupta, Domenic Rosati, and Subhabrata Majumdar. Semantic consistency for assuring reliability of large language models. arXiv preprint arXiv:2308.09138, 2023.

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J Liu. Out-of-distribution detection and selective generation for conditional language models. In The Eleventh International Conference on Learning Representations, 2022.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. arXiv preprint arXiv:2112.08633, 2021.

Soumya Sanyal, Tianyi Xiao, Jiacheng Liu, Wenya Wang, and Xiang Ren. Are machines better at complex reasoning? unveiling human-machine inference gaps in entailment verification. arXiv e-prints, pp. arXiv–2402, 2024a.

Soumya Sanyal, Tianyi Xiao, Jiacheng Liu, Wenya Wang, and Xiang Ren. Minds versus machines: Rethinking entailment verification with language models. arXiv preprint arXiv:2402.03686, 2024b.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538, 2017.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. arXiv preprint arXiv:2309.15789, 2023.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615, 2022.

Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri, and Josep Torrellas. Towards greener llms: Bringing energy-efficiency to the forefront of llm inference. arXiv preprint arXiv:2403.20306, 2024.

Mirac Suzgun and contributors. Big-bench-hard: Challenging big-bench tasks and whether chain-of-thought can solve them. https://github.com/suzgunmirac/BIG-Bench-Hard, 2023. Accessed: 2024-09-16.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261, 2022.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. arXiv preprint arXiv:1811.00937, 2018.

The Mosaic Research Team. Introducing dbrx: A new state-of-the-art open llm, 2024. URL https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm. The Mosaic Research Team.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. Fusing models with complementary expertise. arXiv preprint arXiv:2310.01542, 2023a.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. arXiv preprint arXiv:2406.04692, 2024.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. arXiv preprint arXiv:2305.17926, 2023b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35:24824–24837, 2022.

Minghao Wu and Alham Fikri Aji. Style over substance: Evaluation biases for large language models. arXiv preprint arXiv:2307.03025, 2023.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. arXiv preprint arXiv:2304.12244, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems, 36, 2024.

Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. arXiv preprint arXiv:2305.16582, 2023.

Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. Flask: Fine-grained language model evaluation based on alignment skill sets. arXiv preprint arXiv:2307.10928, 2023.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. arXiv preprint arXiv:2210.03493, 2022.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. Advances in Neural Information Processing Systems, 36, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. arXiv preprint arXiv:2205.10625, 2022.

Zhanghao Zhouyin and Ding Liu. Understanding neural networks with logarithm determinant entropy estimator. arXiv preprint arXiv:2105.03705, 2021.

Appendix

Table of Contents

A Reproducibility Statement

We provide details on the experimental setup, hyperparameters, and data preprocessing steps to maximize the reproducibility of our results. The codebase can be found at: https://anonymous.4open.science/r/balancing_act-CB59/.

Listing 1 - ARC CoT prompt

Q: George wants to warm his hands quickly by rubbing them. Which skin surface will produce the most heat?
Answer Choices:
(a) dry palms
(b) wet palms
(c) palms covered with oil
(d) palms covered with lotion
A: The dry palms will produce the most heat when rubbed together because they create the highest friction, allowing for more efficient conversion of mechanical energy into thermal energy without the lubricating effects of moisture or other substances. The answer is (a).

Q: Michael learned that the movement of Earth in the solar system causes changes that can be seen on the planet. Which change could be seen on Earth in the time it takes Earth to rotate once on its axis?
Answer Choices:
(a) day becoming night
(b) winter changing to spring
(c) January changing to February
(d) a new moon becoming a full moon
A: Day becoming night is the change that can be observed on Earth during one complete rotation on its axis because this rotation is what causes the cycle of daylight and darkness, with different parts of the planet facing towards or away from the Sun over the course of approximately 24 hours. The answer is (a).

Q: Which would a scientist use in trying to model the cause of planetary years?
Answer Choices:
(a) planetary mass
(b) planetary color
(c) a planet's core temperature
(d) a planet's distance from the Sun
A: A planet's distance from the Sun is the key factor a scientist would use to model the cause of planetary years because the length of a planet's orbit around the Sun, which determines its year, is primarily dependent on its distance from the Sun according to Kepler's laws of planetary motion. The answer is (d).

Q: Which phrase does not describe asexual reproduction in organisms?
Answer Choices:
(a) requires two parents
(b) little variation in offspring
(c) only one type of cell involved
(d) duplicates its genetic material
A: 'Requires two parents' does not describe asexual reproduction because asexual reproduction is characterized by a single organism producing genetically identical offspring without the need for a mate, unlike sexual reproduction which involves genetic material from two parents. The answer is (a).

Listing 2 - Aggregation and Synthesis Prompt

You have been provided with a set of responses from various open-source models to the latest user query. Your task is to synthesize these responses into a single, high-quality response in order to answer the final question given to you - note that, in order to help you understand the task, the user may first show some example questions. It is crucial to critically evaluate the information provided in these responses, recognizing that some of it may be biased or incorrect. Your response should not simply replicate the given answers but should offer a refined, accurate, and comprehensive reply to the instruction. Ensure your response is well-structured, coherent, and adheres to the highest standards of accuracy and reliability. If you are asked to provide a single answer to a question, do not include multiple answers in your response.

Responses from models:

<Model 1 Response>
<Model 2 Response>
...
<Model N Response>

B Gated Mixture of Agents Experiment

B.1 Tasks and datasets

We considered the following tasks and their associated benchmarks for the EigenDivergence filtering experiment:

- **Arithmetic reasoning:** We used GSM8K (Cobbe et al., 2021), a set of linguistically diverse grade school math word problems, and MATH (Hendrycks et al., 2021), a set of challenging competition mathematics problems which require step-by-step reasoning to solve.
- **Commonsense reasoning:** For these tasks, we used CommonsenseQA (CSQA) (Talmor et al., 2018), a popular commonsense question answering dataset constructed by extracting target concepts from text that have the same semantic relation to a single source concept; questions then aim to discriminate between the target concepts. Correctly solving the questions often requires prior knowledge. We also used the AI2 Reasoning Challenge (Clark et al., 2018), a dataset of natural grade-school science questions authored for human tests.
- **Instruction following:** We considered AlpacaEval 2.0 (Dubois et al., 2024), which uses a reference-free method (GPT-4) to evaluate the quality of outputs by how aligned they are with human preferences. The benchmark calculates a length-controlled win-rate which explicitly accounts for the confounding whereby models prefer longer answers. This benchmark aligns strongly with human preference and has a Spearman correlation of 0.98 with LMSYS' Chatbot Arena. We also considered MT-Bench (Zheng et al., 2024), a multi-turn question set which uses LLM judges to evaluate answers, and which demonstrates strong agreement with human preferences.

B.2 Language model prompting

We perform all arithmetic and commonsense reasoning tasks in the few-shot setting. For arithmetic reasoning, we generate a 5-shot CoT prompt by randomly selecting samples from the training set of each benchmark for each question. For the CSQA task, we used the same prompt as in (Wang et al., 2022; Wei et al., 2022). For the AI2 Reasoning Challenge tasks, we manually construct a 4-shot CoT prompt which is shown in Listing 1.

We use a similar prompt as in Wang et al. (2024) for the aggregation and synthesis function with two notable exceptions: 1) The addition of an instruction to focus on the 'latest' user query to account for tasks in the few-shot setting, and 2) An instruction to provide a single answer to a query if this is explicitly requested by the user, to account for tasks where there is a single correct answer. The prompt is reproduced in Listing 2.

B.3 Language and embedding models

We used the same LLMs as in the baseline Mixture of Agents ensemble from Wang et al. (2024). Namely, we use Llama-3-70B-Instruct (Touvron et al., 2023), Qwen1.5-72B-Chat (Bai et al., 2023), Qwen1.5-110B-Chat (Bai et al., 2023), Mixtral-8x22B-v0.1 (Jiang et al., 2024), WizardLM-8x22B (Xu et al., 2023), and dbrx-instruct (The Mosaic Research Team, 2024). We used OpenAI's text-embedding-3-small model (OpenAI, 2024a) as the sentence-embedding function $e(\cdot)$ (described in more detail in Sec. 4.1).

B.4 Sampling scheme

Wang et al. (2022) had demonstrated that self-consistency decoding is robust to both sampling strategy and model scale. In view of this, to generate LLM outputs we used temperature sampling (T) (Ackley et al., 1985; Ficler & Goldberg, 2017), with $T = 0.7$ for all LLMs across all experiments. We do not consider top-k truncation (Radford et al., 2019; Fan et al., 2018; Holtzman et al., 2018) or nucleus sampling (Holtzman et al., 2019). This aligns our sampling scheme with prior work on LLM ensembling (Wang et al., 2024; 2022).

B.5 GMoAs and MoEs

The mixture of experts (MoE) approach leverages the relative expertise of specialised modules at inference time to improve performance over monolithic decoder-only autoregressive transformers (Shazeer et al., 2017). An MoE is usually a stack of layers where each layer contains a set of N experts ($\mathcal{E}(\cdot)$) alongside a gating network $\mathcal{G}(\cdot)$ and a residual connection. An MoE produces an output for layer i as

$$x_{i+1} = \sum_{n=1}^{N} \mathcal{G}_{i,n}(x_i)\mathcal{E}_{i,n}(x_i) + x_i. \tag{13}$$

As noted by Wang et al. (2024), the MoA extends the MoE approach to the model level, operating at the prompt interface for multiple 'expert' LLMs without modifying internal activations. However, the MoA approach consolidates the roles of the gating and expert functions for each LLM. In the Gated Mixture of Agents experiment (Sec. 4.1), we explicitly factorize these functions once more as

$$x_{i+1} = \sum_{n=1}^{N} \mathcal{G}_{i,n}(\boldsymbol{O}_i)\mathcal{E}_{i,n}(x_i) + x_i, \tag{14}$$

where $\boldsymbol{O}_i = \sum_n \mathcal{E}_{i,n}(x_i)$, and thus we explicitly gate outputs depending on how divergent they are relative to the other LLM outputs. In a later ablation experiment (Sec. 4.3) we reverse the gating mechanism and remove the most consistent outputs, rather than the most divergent ones.

C Mixture optimization experiment

In the mixture optimization experiment, the setup is the same as for the Gated Mixture of Agents experiment (Appendix B); namely the same tasks and datasets, language model prompting strategies, language and embedding models, and sampling scheme are used. Each baseline heterogeneous mixture of agents is optimized against one of: 1) Arithmetic reasoning; 2) Commonsense reasoning; 3) Instruction following. Optimizations are done in accordance with Alg. 1, which has the following stopping criteria: 1) If no improvement in target benchmark performance (past a manually-selected threshold) in three iterations, halt; 2) If both adding and removing a model to the mixture degrades performance, halt; 3) If the mixture contains a self-ensemble, and this represents an improvement in performance over a heterogeneous ensemble, halt.

```
Data: Initial mixture of models $\{m_k\}$ with usage proportions $U_0(m_k)$
Result: Optimized mixture composition
Initialize mixture composition with models $\{m_k\}$ and usage proportions $U_0(m_k)$;
Run the ensemble with current mixture composition and obtain performance $P_0$;
Measure usage proportions $U_0(m_k)$;
Run the ensemble again with the same mixture composition and obtain performance $P_1$;
Measure usage proportions $U_1(m_k)$;
Set iteration counter $i \leftarrow 1$;
repeat
    Compute performance difference: $\Delta P = P_i - P_{i-1}$;
    foreach model $m_k$ do
        Compute usage difference: $\Delta U(m_k) = U_i(m_k) - U_{i-1}(m_k)$;
    end
    Compute total usage change: $T = \sum_k |\Delta U(m_k)|$;
    foreach model $m_k$ do
        Compute model delta: $\Delta M(m_k) = \left(\frac{\Delta U(m_k)}{T}\right) \times \Delta P$;
    end
    Identify model $m_{\text{min}}$ with the lowest $\Delta M(m_k)$;
    Identify model $m_{\text{max}}$ with the highest $\Delta M(m_k)$;
    Replace model $m_{\text{min}}$ with a clone of model $m_{\text{max}}$;
    Run the ensemble with updated mixture composition and obtain performance $P_{i+1}$;
    Measure usage proportions $U_{i+1}(m_k)$;
    Set $i \leftarrow i + 1$;
until stopping criterion is met;
```

Algorithm 1: Mixture optimization experiment algorithm.

D ABLATION STUDY FOR POST-PROCESSING FUNCTIONS IN ENSEMBLES

In this section we describe the experimental setup for an ablation study whereby we assess three different post-processing functions for LLM ensembles. We assess the 'aggregation and synthesis' function proposed by Wang et al. (2024), simple LLM-ranking (Wang et al., 2022; 2024), and 'universal self-consistency', recently proposed by Chen et al. (2024b). Descriptions of these functions are as follows:

- **Aggregate and synthesize:** This function allows the final LLM in the ensemble to look through all information in the previous layer's outputs, perform a *critical appraisal*, including assessing for bias or incorrect results, and produce a *novel* output, based on the previous outputs.
- **LLM-ranking:** Here, the final LLM is instructed to look through all information the previous layer's outputs, and to simply select the answer it believes to be the best. The function therefore outputs a single, *previously generated* response that the final LLM believes represents the highest quality response.
- **Universal self-consistency (USC):** A recently proposed (Chen et al., 2024b) generalization of self-consistency decoding (Wang et al., 2022). Here, the final LLM looks through all information in the previous layer's outputs, and selects the most highly *repeating facts of information* (thus selects the most consistent response). This is distinguished from aggregation and synthesis by the fact that USC does not perform any sort of critical appraisal/error assessment.

Listings 3, 4, and 5 illustrate the aggregate and synthesize, LLM-ranking, and USC functions, respectively. We assess already 'optimized' LLM ensembles on tasks for which they are already performant. For each run, we only replace the head function. `Qwen1.5-72B` it used as the final LLM in the ensemble across all models (Qwen Team, 2024).

Listing 3 - Aggregation and synthesis function

‘You have been provided with a set of responses from various open-source models to the latest user query. Your task is to synthesize these responses into a single, high-quality response in order to answer the final question given to you - note that, in order to help you understand the task, the user may first show some example question and answer pairs. It is crucial to critically evaluate the information provided in these responses, recognizing that some of it may be biased or incorrect. Your response should not simply replicate the given answers but should offer a refined, accurate, and comprehensive reply to the instruction. Ensure your response is well-structured, coherent, and adheres to the highest standards of accuracy and reliability. If you are asked to provide a single answer to a question, do not include multiple answers in your response.
Responses from the models: ... ’

Listing 4 - LLM ranking function

‘You have been provided with a set of responses from various open-source models to the latest user query. Your task is to evaluate the responses and select the best answer to the final question given by the user - note that, in order to help you understand the nature of the task, the user may first show you some example question and answer pairs. Your response should simply replicate the answer which you believe represents the best response to the final user query, with no further additions of any kind. This should be the only output you produce. If you are asked to provide a single answer to a question, do not include multiple answers in your response.
Responses from the models: ... ’

Listing 5 - Universal self-consistency

‘You have been provided with a set of responses from various open-source models to the latest user query. Your task is to evaluate the responses and select the most consistent response based on majority consensus to the final question given by the user. If there are not consistent reponses, respond by saying ’I cannot answer this query as there is no consistent information in the outputs I have received’ - note that, in order to help you understand the nature of the task, the user may first show you some example question and answer pairs, but you should focus on selecting the most consistent answer based on majority consensus for the final user query.
Responses from the models: ... ’

E Dynamic mixture of agents experiment

In this section we describe the experimental setup for the Dynamic Mixture of Agents (DMoA) experiment on the Big Bench Hard (BBH) benchmark. BBH is a subset of 23 challenging tasks from the BigBench dataset (Srivastava et al., 2022). The tasks include algorithmic reasoning, multi-step arithmetic, and world knowledge (Suzgun et al., 2022; Hugging Face, 2023).

E.1 Corrected BBH

The BBH repository on github (Suzgun & contributors, 2023) contains a number of formatting errors across some tasks. We concatenate all questions across the 23 subsets, and correct for any errors in the expected output format. Listing 6 shows the expected output format for each subset. Listing 7 demonstrates manual corrections of previously incorrectly formatted questions.

Listing 6 - BBH expected output format

The expected output format for each subset is shown below:

- **boolean_expressions**: Binary (True/False)
- **causal_judgement**: Binary (Yes/No)
- **date_understanding**: Multiple-choice (e.g., (A), (B), etc.)
- **disambiguation_qa**: Multiple-choice (e.g., (A), (B), etc.)
- **dyck_languages**: Bracket sequences (e.g., [ ],] [)
- **formal_fallacies**: Binary (valid/invalid)
- **geometric_shapes**: Multiple-choice (e.g., (A), (B), etc.)
- **hyperbaton**: Multiple-choice (e.g., (A), (B), etc.)
- **logical_deduction_five_objects**: Multiple-choice (e.g., (A), (B), etc.)
- **logical_deduction_seven_objects**: Multiple-choice (e.g., (A), (B), etc.)
- **logical_deduction_three_objects**: Multiple-choice (e.g., (A), (B), etc.)
- **movie_recommendation**: Multiple-choice (e.g., (A), (B), etc.)
- **multistep_arithmetic_two**: Numeric (integers or real numbers)
- **navigate**: Binary (Yes/No)
- **object_counting**: Numeric (integers or real numbers)
- **penguins_in_a_table**: Multiple-choice (e.g., (A), (B), etc.)
- **reasoning_about_colored_objects**: Multiple-choice (e.g., (A), (B), etc.)
- **ruin_names**: Multiple-choice (e.g., (A), (B), etc.)
- **salient_translation_error_detection**: Multiple-choice (e.g., (A), (B), etc.)
- **snarks**: Multiple-choice (e.g., (A), (B), etc.)
- **sports_understanding**: Binary (Yes/No)
- **temporal_sequences**: Multiple-choice (e.g., (A), (B), etc.)
- **tracking_shuffled_objects_five_objects**: Multiple-choice (e.g., (A), (B), etc.)
- **tracking_shuffled_objects_seven_objects**: Multiple-choice (e.g., (A), (B), etc.)
- **tracking_shuffled_objects_three_objects**: Multiple-choice (e.g., (A), (B), etc.)
- **web_of_lies**: Binary (Yes/No)
- **word_sorting**: Free-text (open-ended answers)

Listing 7 - Corrections of previously incorrectly formatted questions in BBH

The corrections are shown in the following format: question id: question, corrected answers, target answer:

- **4182**: Which of the following is a humorous edit of this artist or movie name: 'earth, wind, & fire'?
  Options:
  (A) eareth, wind, & fire
  (B) earth, bind, & fire
  (C) earthm, wind, & fire
  (D) dearth, wind, & fire
  Answer: (D)
- **2850**: Find a movie similar to Minority Report, Shrek, Catch Me If You Can, Aladdin:
  Options:
  (A) Monsters, Inc
  (B) Children of the Night
  (C) The Incredible Shrinking Man
  (D) Town & Country
  Answer: (A)
- **4227**: Which of the following is a humorous edit of this artist or movie name: 'rita, sue and bob too'?
  Options:
  (A) rita, sue and bob too
  (B) rita, sue and bob poo
  (C) rita, sue and box too
  (D) ritay sue and bob too
  Answer: (B)

E.2 DMoA PRE-PROCESSING FUNCTION

As described in Sec. 4.4, we require a pre-processing function which takes a user query as its argument and identifies both a set of skills required to produce a high-quality answer to the query, and a prediction of which model(s) are likely to perform well given the prerequisite skills required. An ensemble is then constructed and executed at test-time to answer the query. We use `openai/gpt-4o` (OpenAI, 2024c) as the pre-processing function. Listing 8 shows the prompt for this task. Listing 9 shows an example output for a randomly-selected prompt.

Listing 8 - Pre-processing function for the Dynamic Mixture of Agents

- `<general information>`
  You will be shown performance information on a number of benchmarks for the following reference models: microsoft/WizardLM-2-8x22B, Qwen/Qwen1.5-110B-Chat, Qwen/Qwen2-72B-Instruct, meta-llama/Llama-3-70b-chat-hf, mistralai/Mixtral-8x22B-Instruct-v0.1, databricks/dbrx-instruct
  `</general information>`

- `<benchmark descriptions>`
  IFEval: IFEval is a dataset designed to test a model's ability to follow explicit instructions, such as "include keyword x" or "use format y." The focus is on the model's adherence to formatting instructions rather than the content generated, allowing for the use of strict and rigorous metrics.

  MATH: MATH is a compilation of high-school level competition problems gathered from several sources, formatted consistently using Latex for equations and Asymptote for figures. Generations must fit a very specific output format. We keep only level 5 MATH questions and call it MATH Lvl 5.

  GPQA (Graduate-Level Google-Proof Q&A Benchmark): GPQA is a highly challenging knowledge dataset with questions crafted by PhD-level domain experts in fields like biology, physics, and chemistry. These questions are designed to be difficult for laypersons but relatively easy for experts. The dataset has undergone multiple rounds of validation to ensure both difficulty and factual accuracy.

  MuSR (Multistep Soft Reasoning): MuSR is a new dataset consisting of algorithmically generated complex problems, each around 1,000 words in length. The problems include murder mysteries, object placement questions, and team allocation optimizations. Solving these problems requires models to integrate reasoning with long-range context parsing.

  MMLU-PRO (Massive Multitask Language Understanding - Professional): MMLU-Pro is a refined version of the MMLU dataset, which has been a standard for multiple-choice knowledge assessment. Recent research identified issues with the original MMLU, such as noisy data (some unanswerable questions) and decreasing difficulty due to advances in model capabilities and increased data contamination. MMLU-Pro addresses these issues by presenting models with 10 choices instead of 4, requiring reasoning on more questions, and undergoing expert review to reduce noise.
  `</benchmark descriptions>`

- `<model benchmark performance>`

LLM Name	IFEval	MATH Lvl 5	GPQA	MUSR	MMLU-PRO
Qwen/Qwen1.5-110B-Chat	59.39	0	12.19	16.29	42.50
Qwen/Qwen2-72B-Instruct	38.24	29.15	19.24	19.73	NA
microsoft/WizardLM-2-8x22B	52.72	22.28	17.56	14.54	39.96
meta-llama/Llama-3-70b-chat-hf	80.99	23.34	4.92	10.92	46.74
mistralai/Mixtral-8x22B-Instruct-v0.1	71.84	18.73	16.44	13.49	38.70
databricks/dbrx-instruct	54.16	6.87	12.19	12.20	29.81

  `</model benchmark performance>`

- `<your task>`
  I'm going to show you a prompt (user query). Note that to help you better understand the nature of the task, you may be given a chain of thought (3-shot CoT) prompt, and you'll be expected to focus on the final query. I'd like you to perform the following functions:
  First, identify a list of skills you think are relevant for producing a high-quality response to this query.
  Second, predict which models given the `<general information>` and `<model benchmark performance>` are best suited to answer the required query.
  Finally, produce a list of 6 choices to create an ensemble of the models. This list can include repeating model choices. The list should be of the format:

  ”model_name_1,model_name_2, ..., model_name_n”

  There should be no spaces between the names, and no other punctuation or output must be produced. The names must match the entire string given in the general information section exactly. For instance, given the MATH Lvl 5 scores, supposing a task was mathematical in nature, then you may wish to create a list such as:

  ”Qwen/Qwen2-72B-Instruct,Qwen/Qwen2-72B-Instruct,Qwen/Qwen2-72B-Instruct,Qwen/Qwen2-72B-Instruct,meta-llama/Llama-3-70b-chat-hf,meta-llama/Llama-3-70b-chat-hf”

  This has 6 models, and it contains two models which have the highest MATH Lvl 5 performance. One is repeated 4 times, and the other twice. This is just an illustrative example. I'm going to show you the actual prompt/query, and would like you to carry out this task.
  `</your task>`

Listing 9 - Example outputs for LLM-based pre-processing function

- **Prompt:**
  How would a typical person answer each of the following questions about causation? Eugene and Tina were a young married couple who lived in the country. Both were partially paralyzed and confined to wheelchairs. They had met four years before when Tina was a counsellor with the Canadian Paraplegic Association, had fallen in love, and were married one year later. On this particular evening, Eugene had phoned to request a cab to take them downtown. When the taxi driver arrived, Eugene and Tina were waiting by the street. On seeing that they were both in wheelchairs, the taxi driver refused their fare because he thought it would be too crowded in the taxi with both of them and the wheelchairs. So the taxi driver headed back downtown without them. Because there was no time to call another cab, Eugene and Tina took Tina's car, which was equipped with special hand controls. In order to get downtown from their house, they had to travel across a bridge over Rupert River. A severe storm the night before had weakened the structure of the bridge. About 5 minutes before Eugene and Tina reached it, a section of the bridge collapsed. The taxi driver had reached the bridge shortly before them, and had driven off the collapsed bridge. He barely managed to escape from his taxi before it sank in the river. In the dark, Eugene and Tina drove off the collapsed bridge and their car plummeted into the river below. They both drowned. Their bodies were retrieved from the car the next morning. Did the taxi driver's refusal to take Eugene and Tina cause their death?
  Options:
  - Yes
  - No

- **Skills relevant to the query:**
  - **Causal reasoning**: The query involves determining the chain of events and their contribution to the final outcome (the deaths of Eugene and Tina).
  - **Ethical and moral reasoning**: The scenario requires consideration of responsibility and potential negligence.
  - **Legal reasoning**: The question is somewhat legalistic, as it asks about causation and fault, which are common in legal contexts.
  - **Long-context understanding**: The model must handle a lengthy narrative and parse multiple factors in the scenario.
  - **Commonsense reasoning**: Determining how a typical person would respond requires an understanding of everyday logic and reasoning.
- **Best-suited models based on the benchmarks:**
  - **Multistep soft reasoning (MuSR)** and **Instruction following (IFEval)** are key benchmarks for this type of task. These scores measure reasoning through complex scenarios and adherence to following instructions.
  - **Meta-llama/Llama-3-70b-chat-hf**: Strong **IFEval** score (80.99) and moderate scores in reasoning-based tasks (MuSR: 10.92).
  - **Mistralai/Mixtral-8x22B-Instruct-v0.1**: Strong **IFEval** (71.84) and reasonable MuSR performance (13.49).
  - **Microsoft/WizardLM-2-8x22B** and **Qwen/Qwen2-72B-Instruct** also have decent MuSR and IFEval scores, making them good candidates for causal reasoning and instruction-following tasks.
- **Ensemble of 6 models:**
  'meta-llama/Llama-3-70b-chat-hf,meta-llama/Llama-3-70b-chat-hf,mistralai/Mixtral-8x22B-Instruct-v0.1,mistralai/Mixtral-8x22B-Instruct-v0.1,microsoft/WizardLM-2-8x22B,Qwen/Qwen2-72B-Instruct'

Table 3: Performance of the DMoA across instruction following, arithmetic reasoning, and commonsense reasoning benchmarks. The DMoA outperforms other models and ensembling strategies across the majority of the benchmarks. † denotes our replication of results.

	Instruction Following		Arithmetic Reasoning		Commonsense Reasoning		
Model	**AlpacaEval 2.0**	**MT-Bench**	**GSM8K**	**MATH**	**CSQA**	**ARC-C**	**ARC-E**
DMoA	**63.21**	**9.19**	**96.67**	**71.23**	87.51	92.50	**94.47**
GMoA	58.66	8.97	94.23	56.35	85.20	92.32	93.75
MoA	59.50	9.19	93.87	55.22	84.32	91.85	94.31
Llama-3-70B	34.4	8.8	93.0	50.4	83.8	90.5†	94.1
Qwen-1.5-110B	43.9	8.9	85.4	49.6	82.1†	69.6	93.9†
Qwen-1.5-72B	36.6	8.4	79.5	34.1	83.2†	65.9	92.7†
WizardLM-8x22B	51.3	8.8	81.6	22.7	69.0†	62.5	90.1†
Mixtral 8x22B	30.9	8.8	83.7	41.7	81.7†	70.7	91.8†
DBRX-Instruct	25.4	8.4	72.8	32.5	82.2†	68.9	89.7†
GPT-4 Omni (05/13)	57.5	9.19	94.1†	61.2†	**88.6**†	**94.6**†	94.3†

E.3 BBH EVALUATION

Whilst the tasks in BBH all use objective metrics, we found that regex patterns can be brittle to correctly extracting the answer produced by a model. As a result, we found that BBH performance can be underestimated by using regex-based answer-extraction approaches in a pilot study. Based on the observation by Chen et al. (2024b) that LLMs more easily compute consistency between responses that judging the correct answer to a question directly, we therefore use an LLM-as-a-judge approach to evaluate BBH performance. Namely, we use `openai/gpt-3.5-turbo` (OpenAI, 2024b) to evaluate the model responses. We first extract the answer from the proposed output and then perform entailment verification (Sanyal et al., 2024b) between the extracted answer and the ground-truth answer. Listing 10 illustrates the prompt for this.

Listing 10 - Answer extraction and entailment approach for BBH

- **Query:** {example['input']}
- **Prediction:** {example['output']}
- **Ground Truth:** {example['target']}

Does the prediction match the provided ground truth?
First extract the final answer from the prediction, then, see if it matches the provided ground truth, and then answer with 'yes' or 'no'.

We additionally note the large discordance between quoted BBH scores. For instance, the `Qwen1.5-110B` model is cited as having a BBH score of 74.8 (Qwen Team, 2024). However on the HuggingFace (HF) open LLM leaderboard, the same model has a score of 44.28 (Open LLM Leaderboard Team, 2024). We believe this is partially due to HF quoting normalized accuracy, whilst several labs quote raw accuracy. To ensure fair comparison, we therefore assess all models and ensembles using our BBH evaluation pipeline, and quote raw accuracy as well as normalized accuracy. For all models/mixtures, we also note the best and worst performing subsets.

E.4 ADDITIONAL RESULTS

We additionally report the perform of DMoA on the instruction following, arithmetic reasoning, and commonsense reasoning benchmarks used for the gated mixture of agents and mixture optimization experiments. As can be seen in Table. 3, the DMoA framework outperforms other models and ensembling strategies across the majority of the benchmarks, further supporting our results in the BBH experiment.

F Concordance between internal and projected embeddings

This section describes an experiment to demonstrate the relationship between eignscores calculated from the internal token embedding space of a LLM, and eigenscores calculated from projecting natural language outputs into a shared sentence embedding space.

F.1 Experimental setup

For the internal Eigenscore we use the final token embedding of the middle layer of `llama-3-8B` (Touvron et al., 2023) as per Chen et al. (2024a). We sample 100 datapoints from AlpacaEval 2.0 and for each question sample 4 generations with a temperature of 0.3 and a top-k of 50. The length and repitition penalty parameters are both set to 1. For the 'external' eigenscore, we project the natural language outputs from each generation into a semantic embedding space using one of three models: OpenAI's `text-embedding-3-small`, `text-embedding-3-large`, and `text-embedding-ada-002` (OpenAI, 2024) - we then perform the Eigenscore calculation on these embeddings and calculate correlations for scores derived from each set of 'external' embeddings with Llama's internal embeddings.

F.2 Results

Results are illustrated in Fig. 5. As can be seen, there is a strong positive relationship between Eignscores calculated from the internal token embedding space of `llama-3-8B` and those calculated from first projecting the natural language output to a sentence embedding space using `text-embedding-3-small`, `text-embedding-3-large`, or `text-embedding-ada-002`. The Pearson's correlation coefficients of were 0.731, 0.779, and 0.781 (3 s.f.), respectively, demonstrating minimal variance and robust concordance for different text embeddings models.

G On the relationship between EigenDivergence and information gain

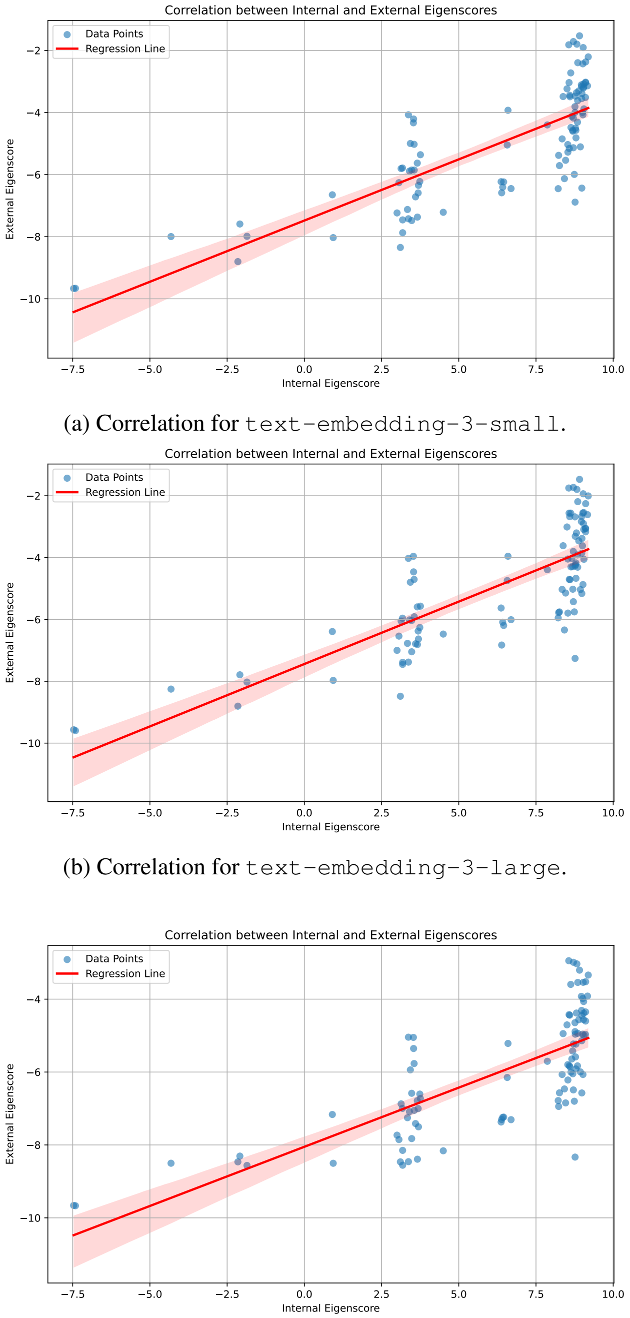
In machine learning, information gain is often used to refer to the mutual information $I(\cdot)$ between two random variables (X, Y) as

$$I(X;Y) := D_{\text{KL}}(P_{X,Y}||P_X \otimes P_Y), \tag{15}$$

which can equivalently be expressed as a difference of entropies as follows:

$$\begin{aligned}
I(X;Y) &= \int_{\mathcal{Y}}\int_{\mathcal{X}} P_{X,Y}(x,y)\log\left(\frac{P_{X,Y}(x,y)}{P_X(x)P_Y(y)}\right)dxdy \\
&= \int_{\mathcal{Y}}\int_{\mathcal{X}} P_{X,Y}(x,y)\log\left(\frac{P_{X,Y}(x,y)}{P_X(x)}\right)dxdy + \int_{\mathcal{Y}}\int_{\mathcal{X}} P_{X,Y}(x,y)\log\frac{1}{P_Y(y)}dxdy \\
&= \int_{\mathcal{Y}}\int_{\mathcal{X}} P_{X,Y}(x,y)\log\left(\frac{P_{X,Y}(x,y)}{P_X(x)}\right)dxdy - \int_{\mathcal{Y}}\int_{\mathcal{X}} P_{X,Y}(x,y)\log P_Y(y)dxdy \\
&= \int_{\mathcal{X}} P_X(x)\left(\int_{\mathcal{Y}} P_{Y|X=x}(y)\log P_{Y|X=x}(y)dy\right)dx \\
&\quad - \int_{\mathcal{Y}}\left(\int_{\mathcal{X}} P_{X,Y}(x,y)dx\right)\log P_Y(y)dy \\
&= -\int_{\mathcal{X}} P_X(x)H(Y|X=x)dx - \int_{\mathcal{Y}} P_Y(y)\log P_Y(y)dy \\
&= -H(Y|X) + H(Y) \\
&= H(Y) - H(Y|X),
\end{aligned}$$

Where $H(Y)$ is the marginal entropy of random variable Y and $H(Y|X)$ is the conditional entropy of Y given X. Unless (X, Y) are independent, then by learning or assuming the information in X, we can measure the change in uncertainty about Y, and therefore information gain measures



(a) Correlation for `text-embedding-3-small`.

(b) Correlation for `text-embedding-3-large`.

(c) Correlation for `text-embedding-ada-002`.

Figure 5: Correlation between internal token embeddings from `llama-3-8B` and external semantic embeddings from `text-embedding-3-small`, `text-embedding-3-large`, and `text-embedding-ada-002`. The Pearson's correlation coefficients of are 0.730, 0.779, and 0.781, respectively. These coefficients indicate a strong positive relationship, demonstrating the consistency between Eigenscore calculations from internal and projected embedding spaces.

the reduction in entropy when a feature is known. As can be seen in Eq. 9, the EigenDivergence is proportional to the difference in differential entropies in sentence embedding space when comparing all LLM outputs to a subset where one output has been removed. Interpreting $E_{-i}(\boldsymbol{\Sigma}{-}i|\boldsymbol{x};\boldsymbol{\theta}{-}i)$

analogously to $H(Y|X)$ as the entropy of the ensemble given that we exclude the i-th output demonstrates that the EigenDivergence effectively captures the change in uncertainty about the ensemble's prediction space when excluding a specific model output.

H Investigation of the EigenDivergence (ED) score

In this Appendix we further investigate the potential impact of using the EigenDivergence score as a filtering mechanism for GMoA. Namely, we want to assess the correlation between EigenDivergence and accuracy of individual queries prior to the filtering mechanism in order to establish the effectiveness at removing incorrect queries from mixtures.

Our experimental design is as follows; firstly, we construct a single layer GMoA using the Llama-3-70B-Instruct, Qwen2-72B-Instruct, Mixtral-8x22B-v0.1, WizardLM-8x22B, and dbrx-instruct models, using Qwen2-72B-Instruct as the head model. We sub-sample 200 questions from the GSM8K and ARC-C benchmarks respectively, and for each question, we capture the individual responses for a given layer prior to aggregation and calculate and store the EigenDivergence scores for each response.

H.1 Results

Difference in distributions of EigenDivergences categorised by correct and incorrect answers across benchmarks, as well as considering correct answers for a given layer with at least one incorrect answer, are shown in Fig. 6.

2400 individual layer answers were assessed over the GSM8K and ARC-C benchmarks. For the GSM8K sub-sample, 1119 (93%) answers were correct, whilst 81 (6.7%) answers were incorrect. Over the 200 questions, 148 (74%) had 6 correct answers, 39 had (19.5%) 5 correct answers, 4 (2.0%) had 4 correct answers, 5 (2.5%) had 3 correct answers, 2 (1.0%) had 2 correct answers, 1 (0.05%) had 1 correct answer and 1 (0.05%) had 0 correct answers. We find that correct answers had a mean (median) ED score of -0.269 (-0.307) and a standard deviation of 0.263. The inter-quartile range for correct answers was found to be [-0.439, -0.166]. For incorrect answers, the mean (median) ED score was found to be -0.065 (-0.168) with a standard deviation of 0.317. The inter-quartile range for incorrect answers was found to be [-0.288, 0.082]. Considering only correct questions with at least one incorrect answer in a given GMoA layer, we find that correct answers had a mean (median) ED score of -0.314 (-0.332) and a standard deviation of 0.210. The inter-quartile range for correct answers was found to be [-0.447, -0.209].

For the ARC-C sub-sample, 1069 (89.1%) individual layer answers were correct, whilst 131 (10.9%) were incorrect. Over the 200 questions, 134 (67.0%) had 6 correct answers, 39 (19.5%) had 5 correct answers, 12 (6.0%) had 4 correct answers, 5 (2.5%) had 3 correct answers, 2 (1.0%) had 2 correct answers, 3 (1.5%) had 1 correct answers, and 5 (2.5%) had no correct answers. We find that correct answers had a mean (median) ED score of -0.238 (-0.267) and a standard deviation of 0.285. The inter-quartile range for correct answers was found to be [-0.422, -0.101]. For incorrect answers, the mean (median) ED score was found to be -0.152 (-0.185) with a standard deviation of 0.263. The inter-quartile range for incorrect answers was found to be [-0.315, -0.009]. Considering only correct questions with at least one incorrect answer in a given GMoA layer, we find that correct answers had a mean (median) ED score of -0.251 (-0.274) and a standard deviation of 0.266. The inter-quartile range for correct answers was found to be [-0.414, -0.107].

Comparatively, these results show that ED scores of individually correct queries are more negative on average than individually incorrect queries. Additionally, in both datasets, we find that correct answers in layers with at least one incorrect query are more negative on average. This suggests that filtering based on ED scores is more likely to lead to removing diverse and potentially incorrect answers for close-ended tasks such as arithmetic and commonsense reasoning.

H.2 Qualitative Analysis

Monthly cell phone costs We demonstrate a failure mode of the ED score with the following question from the GSM8K dataset: *'Shiela bought five cell phones for $150 each for a 3-month installment. A 2% interest will be charged for each unit. How much will Shiela*

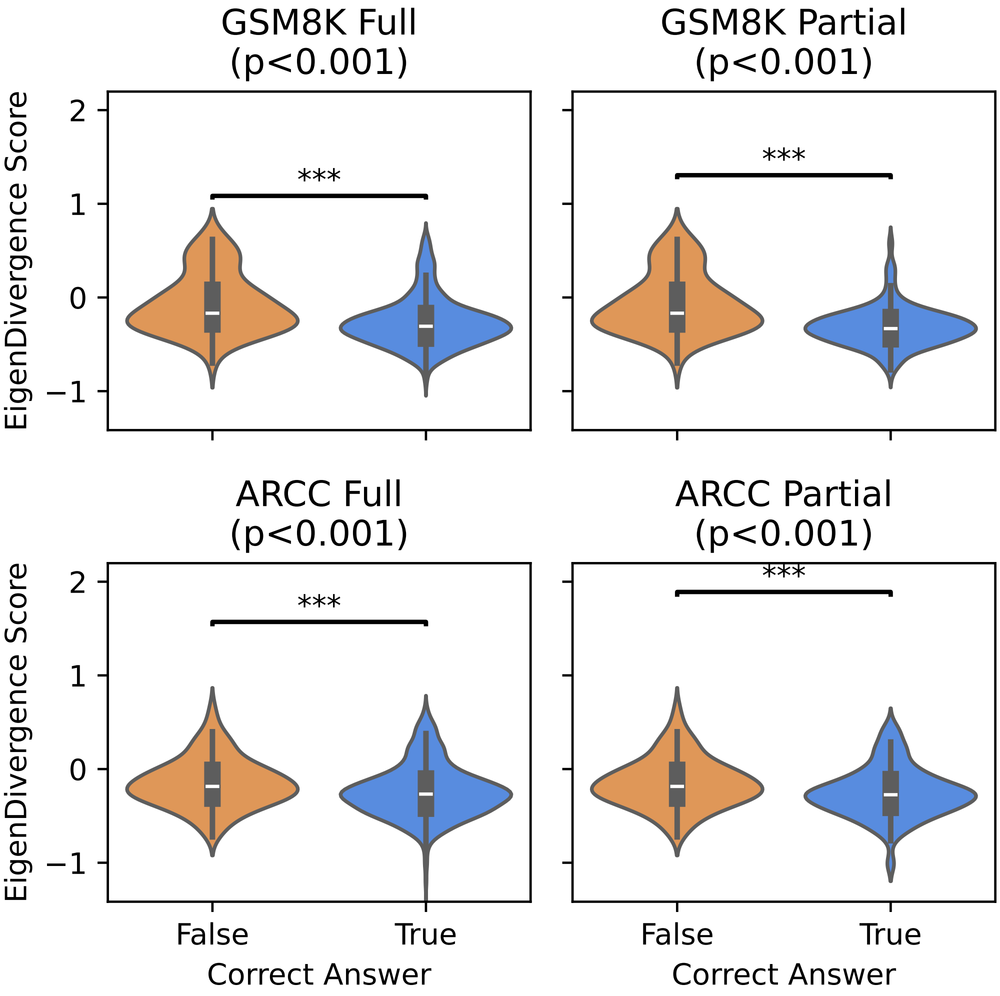


Figure 6: **Distribution of EigenDivergence scores with answer correctness across GSM8K and ARCC subsets.** EigenDivergence scores are more positive (i.e. increase semantic diversity if considered in the mixture) for incorrect answers, and conversely more negative (i.e increase semantic consistency in the mixture) when correct. Across all comparisons, Wilcoxon rank-sum tests revealed statistically significant (p<0.001) distributions when comparing correct versus incorrect answers, marked with (***). In the second column, we compare correct answers from questiosn with at least one incorrect answer in the layer; in these cases correct answers are *more* negative.

pay each month for 3 months?'. As can be seen in Table. 4, `dbrx-instruct` and `Mixtral-8x22B-Instruct-v0.1` show a remarkably similar semantic template in their reasoning:

1. 'First, let's calculate the total cost. . . ',
2. 'Next, let's calculate the [total] interest. . . ',
3. 'Finally, let's calculate the [monthly payment/ total amount] Sheila. . . '.

However, both make errors in their calculations and reach different wrong conclusions (of $755 and $265, respectively). Meanwhile, `Llama-3-70b-chat-hf` and `Qwen1.5-110B-Chat` take slightly different approaches from one another to deduce the correct answer. `Llama-3-70b-chat-hf` opts to calculate the total interest in one step before calculating the total cost and then per-monthly cost, whilst `Qwen1.5-110B-Chat` correctly calculates

Table 4: Comparison of model responses for calculating cell phone installment payments.

ED Score	Model	Response
-0.255	meta-llama/Llama-3-70b-chat-hf	Let's break it down step by step! Shiela bought 5 cell phones for $150 each, so the total cost is: 5 x $150 = $750 A 2% interest will be charged for each unit, which means the total interest is: $750 x 2% = $750 x 0.02 = $15 The total amount Shiela needs to pay is: $750 + $15 = $765 Since it's a 3-month installment, Shiela will pay: $765 ÷ 3 = $255 per month for 3 months. #### 255
-0.125	Qwen/Qwen1.5-110B-Chat	5 cell phones * $150 each = $<<5*150=750>>750 Next, calculate the total interest for all cell phones. The interest rate is 2% per unit, so for all five units: 2% of $150 = 0.02 * $150 = $3 per cell phone Total interest for 5 cell phones = 5 * $3 = $<<5*3=15>> Total amount to be paid including interest = Cost of cell phones + Total interest = $750 + $15 = $<<750+15=765>> This total amount is spread over 3 months, so the monthly payment will be: $765 / 3 months = $<<765/3=255>>
-0.376	databricks/dbrx-instruct	First, let's calculate the total cost of the cell phones without interest: 5 cell phones * $150 per cell phone = $750 Next, let's calculate the interest rate per month for one cell phone: 2% per 3 months is approximately 0.67% per month (2/3). 0.67% of $150 is $1 (0.0067 * 150). Now, let's calculate the monthly interest for all 5 cell phones: $1 per cell phone * 5 cell phones = $5 Finally, let's calculate the total amount Shiela will pay each month: $750 (the cost of the cell phones) + $5 (the monthly interest) = $755 So, Shiela will pay $755 each month for 3 months.
-0.319	mistralai/Mixtral-8x22B-Instruct-v0.1	First, let's calculate the total cost of the five cell phones: 5 cell phones * $150 per cell phone = $<<5*150=750>>750 Next, let's calculate the total interest for the 3-month installment period: 2% interest per month * $750 * 3 months = 0.02 * $750 * 3 = $<<0.02*750*3=45>>45 Now, let's add the total interest to the total cost to find out the total amount Shiela has to pay: Total cost + Total interest = $750 + $45 = $<<750+45=795>>795 Finally, let's calculate the monthly payment Shiela has to make for 3 months: Total amount / 3 months = $795 / 3 = $<<795/3=265>>265 per month So, Shiela will pay $265 each month for 3 months.

the interest per unit first, then the total additional costs accrued from the interest, and then the total overall cost and per-monthly cost. By filtering using a semantic consistency score, we remove both correct answers (as the other models use a syntactically near-identical reasoning flow), leading the final aggregator to also answer incorrectly by giving a final answer of $251.11. This failure mode is a likely limitation of all consistency scores which do not explicitly take account of the meaning of the answer being provided (Kadavath et al., 2022; Malinin & Gales, 2020; Lin et al., 2023; Ren et al., 2022; Lin et al., 2022; Chen et al., 2024a; Jesson et al., 2024; Kuhn et al., 2023).

I Cost analysis

We aim to investigate the trade-offs between cost efficiency and performance in the challenging Big Bench Hard benchmark (Suzgun & contributors, 2023) based on our analysis in Sec. 4.4. Fig. 7 illustrates performance against cost. Costs were calculated from the pricing information found on the API providers' websites[1]. The pareto optimal front illustrates models/frameworks which balance cost the performance most effectively. Most individual models offer relatively low-cost options with moderate performance. Ensembles exhibit a clear boost in performance but at increased costs. Frontier models outperform open-source models, however this also comes at a cost premium. As can be seen, the DMoA offers a well balanced option on the pareto front. In particular, it achieves similar performance to gpt-4o-2025-05-13 whilst remaining cheaper to inference. Indeed, the pareto front produced by DMoA → Claude-3.5-Sonnet → DMoA/Sonnet strictly dominates gpt-4o-2025-05-13. Notably DMoA/Sonnet achieves the highest normalized accuracy but is also the most expensive, whilst the fully open-source DMoA provides a more balanced trade-off with high performance at moderate cost.

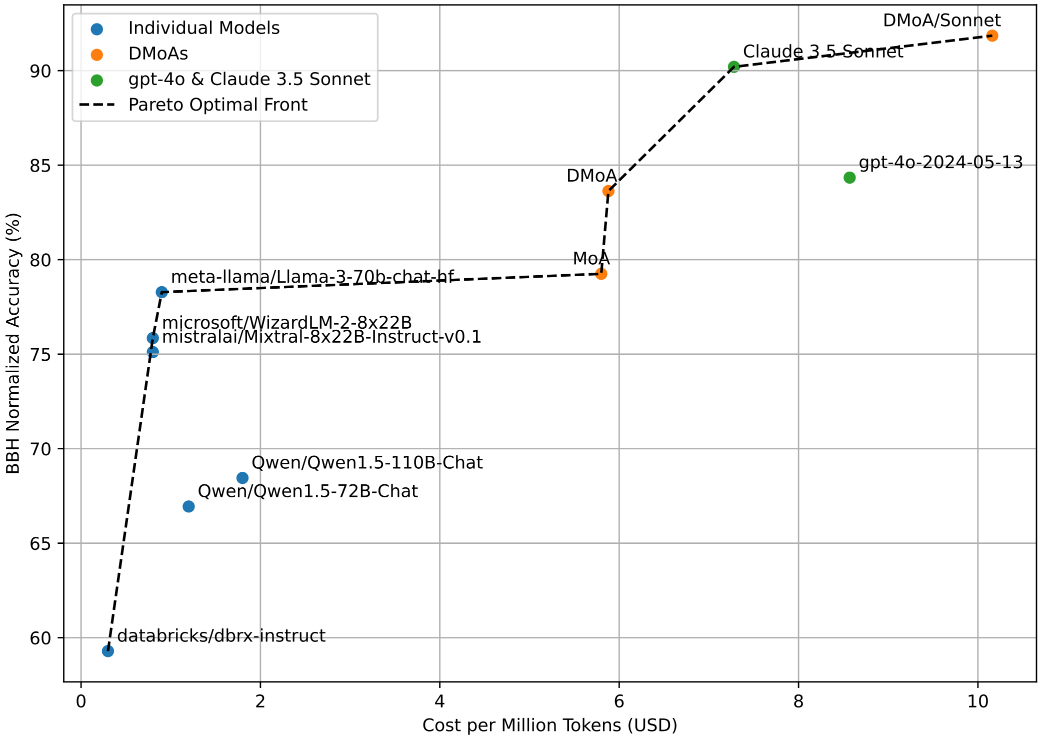
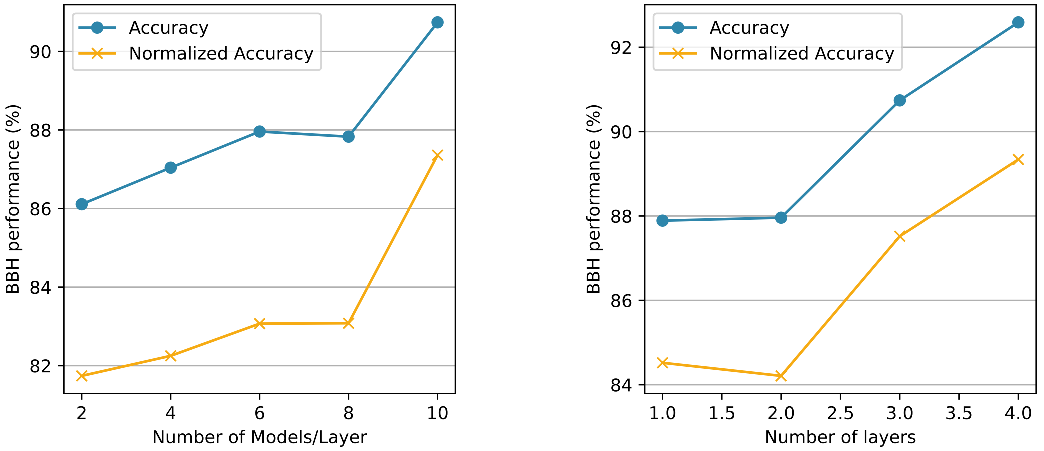


Figure 7: Normalized accuracy versus cost per million tokens for various AI models. The plot illustrates the trade-offs between model performance (normalized accuracy) and operational cost, with individual models, ensemble models (DMoAs), and high-performance standalone models (GPT-4o and Claude 3.5 Sonnet) represented by distinct colors. The dashed line marks the Pareto optimal front, highlighting models that achieve the best balance between cost and accuracy without being strictly dominated by others.

J Test-time scaling

We investigate whether the DMoA framework aligns with test-time scaling laws. Namely, we investigate whether increasing the number of models per layer leads to consistent improvements in

[1] For Together AI: https://www.together.ai/pricing. For anthropic: https://www.anthropic.com/pricing. For OpenAI: https://openai.com/api/pricing/



(a) Scaling the number of models per layer.

(b) Scaling the number of layers in the framework.

Figure 8: **Test-time scaling**. **Left:** Effect of the number of LLMs per layer in the Dynamic Mixture of Agents framework. As more models are introduced in a layer, Big Bench Hard benchmark performance improves. **Right:** The effect on the number of layers in the framework. As more layers are added, the DMoA demonstrates higher performance on the Big Bench Hard benchmark.

performance, and additionally whether scaling the number of layers in the framework is associated with improved performance.

The baseline framework contains 6 models per layer. To investigate fewer models in a given layer, we keep the pre-processing function for the DMoA fixed (as per Appendix E.2), and randomly subselect fewer models from the pool of models predicted to perform well for a given query. Conversely, to increase the number of models in a layer, we sample additional models (with replacement) from the pool of models which are predicted to perform well for the current query. Fig. 8a illustrates that as layer dimension is increased, BBH performance consistently improves.

We investigate scaling the number of layers in the framework whilst fixing the layer dimension (the number of models/layer) to 6. As can be shown in Fig. 8b, adding more layers can also improve BBH performance. In our analysis, increasing the number of layers was associated with a higher final performance than increasing the number of models for a single layer.