DART: Zero-Shot Dual-Side Alignment Routing for LLM Performance-Cost Tradeoffs

Yuejun Jiao **Yanxin Yang** **Boyu Wang** **Yonghao Yang** **Hao Shen** **Mingsong Chen***
MoE Eng. Research Center of HW/SW Co-design Tech. and App., East China Normal University

Abstract

As the ecosystem of Large Language Models (LLMs) rapidly expands, LLM routing has become essential for dynamically balancing performance and cost. However, existing methods fail to simultaneously satisfy the three core criteria of an ideal router, i.e., high routing precision, minimal intrinsic operational overhead, and zero-shot onboarding of new models. These methods rely heavily on surface-level semantics and discrete ID mappings, conflating mere semantic similarity with underlying capability requirements and thereby precluding zero-shot, training-free generalization. To address these limitations, we propose DART, a novel zero-shot routing framework grounded in dual-side alignment to optimize LLM performance-cost tradeoffs. Rather than relying on direct semantic matching, we decouple routing into a demand-supply alignment problem within a shared low-rank latent capability space. Specifically, on the supply side, DART constructs latent capability embeddings for candidate models, using an initialization mechanism to anchor newly introduced models in this latent space via metadata and family-graph priors, without collecting new query–model observations or retraining. On the demand side, a dedicated encoder explicitly distills the query's directional capability requirements and inherent difficulty into a latent demand embedding. This decoupled representation computes routing decisions via a lightweight dot-product operation between the demand and capability embeddings, modulated by query-adaptive cost sensitivity, minimizing overhead. Extensive experiments across comprehensive benchmarks demonstrate that DART achieves state-of-the-art utility under diverse cost constraints, satisfying the three core criteria with exceptional robustness in cold-start scenarios.

1 Introduction

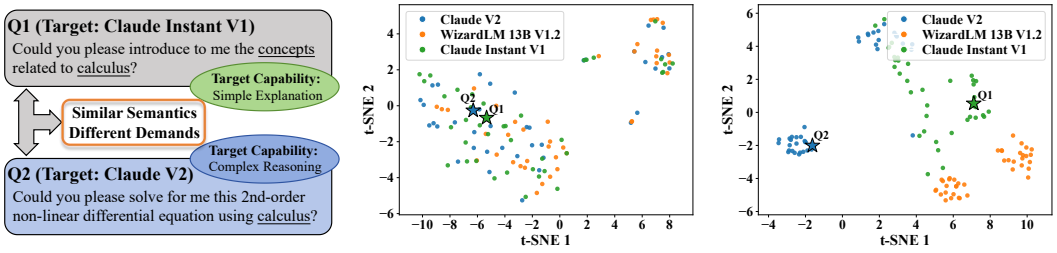
The rapid expansion of large language models (LLMs) has cultivated a diverse ecosystem of models that vary significantly in scale, capabilities, and computational demands. To navigate this diversity, LLM routing has emerged as a critical mechanism that dynamically assigns each user query to the candidate model that balances response performance and inference cost. However, as the ecosystem continues to evolve, the conventional paradigm of routing between only two models (e.g., a strong and a weak LLM) [9, 16] is no longer sufficient for real-world applications. Instead, modern routers must satisfy three core criteria: i) attain high routing precision, ii) minimize their intrinsic operational overhead, and iii) exhibit strong generalization to seamlessly accommodate an ever-expanding pool of candidate models.

Although recent studies have explored various routing strategies, they generally struggle to satisfy these criteria simultaneously. Hybrid LLM [9] and RouteLLM [16] restrict the routing decision to merely a pair of strong and weak models, a paradigm that fails to accommodate the massive scale and diversity of multi-model ecosystems. To handle more complex scenarios, some methods

*Corresponding author: mschen@sei.ecnu.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

employ deep graph neural networks [11] or even use capable LLMs as routers [24], which inevitably incur substantial operational overhead and unacceptable inference latency. Moreover, conventional query-centric routers [22, 29] formulate model selection as a semantic classification problem that maps input text embeddings to discrete LLM IDs, inherently hindering generalization. *Therefore, how to achieve the three core criteria of routers for dynamically expanding LLM ecosystems remains an important challenge.*



(a) The Root Problem (b) Semantic Embedding Space (c) Latent Capability Space

Figure 1: Fundamental limitation of semantic-only routing. (a) Two queries sharing highly similar surface semantics (e.g., both involving calculus) can exhibit drastically different capability demands. (b) In the original semantic space, such queries are severely entangled. (c) In the latent capability space, DART effectively disentangles them and aligns perfectly with the model's capabilities.

The root cause of this challenge lies in the widespread reliance on surface-level semantics and discrete ID mappings. As illustrated in Figure 1a, queries with highly similar surface semantics can entail drastically different demands. Consequently, when relying on conventional semantic encoders, queries that require different target LLMs become severely entangled in the semantic space (Figure 1b). To rescue routing precision from this entanglement, current methods are forced to rely on heavyweight intermediate models, which inherently inflate intrinsic operational overhead. In contrast, explicitly distilling the query's capability requirements reduces the routing decision to a lightweight mathematical computation, thereby eliminating the need for expensive intermediate routing layers. Moreover, because existing paradigms memorize a direct mapping from input semantics to model IDs, they are inherently closed-set. Integrating a newly introduced LLM mandates collecting new routing annotations to iteratively fine-tune the decision head, making zero-shot onboarding unattainable.

To address these limitations, we propose DART, a novel zero-shot routing framework grounded in dual-side alignment to optimize LLM performance-cost tradeoffs. By reformulating the routing problem as a demand-supply alignment problem in a low-rank latent capability space, DART uses a dedicated demand-side encoder to explicitly distill latent demand embeddings, disentangling semantic confusion (Figure 1c) and significantly improving routing precision. On the supply side, instead of relying on discrete ID mappings, we construct an offline performance matrix from readily available execution-independent observations and derive model capability embeddings through low-rank factorization of this matrix. This decoupled dual-side representation allows the final routing decision to be computed via ultra-lightweight dot-product operations between latent demand and model capability embeddings, thereby eliminating the need for expensive intermediate routing layers and minimizing the router's operational overhead. Furthermore, to facilitate cold-start onboarding, we introduce a capability initialization mechanism that uses static model metadata and family-graph priors, enabling the system to seamlessly anchor newly introduced models in a zero-shot manner without retraining. Zero-shot refers specifically to model onboarding after the router has been trained on an existing model pool, and DART remains an offline-supervised router whose initial demand-side training uses query-level observations. This paper makes the following three major contributions:

- We propose DART, which reformulates zero-shot LLM routing as a demand-supply alignment problem in a shared low-rank latent capability space. DART introduces a dedicated demand-side encoder that effectively distills query-specific demand embeddings, overcoming the fundamental limitations of surface-level semantic similarity.
- We design a capability initialization mechanism that facilitates zero-shot onboarding of new models without retraining, enabling decisions via lightweight dot-product operations coupled with query-adaptive cost sensitivity, thereby minimizing operational overhead.
- We conduct extensive experiments on comprehensive routing benchmarks to show that DART achieves state-of-the-art utility across diverse cost constraints, successfully satisfying the three core criteria with exceptional robustness in cold-start scenarios.

2 Related Work

Query-Centric Routing. Query-centric routers formulate routing as a supervised classification from query representations to model selections. Representative methods include KNNRouter, MLPRouter, and MFRouter, which are published in TO-Router [22] and RouteLLM [16]. KNNRouter performs nearest-neighbor retrieval over query embeddings, MLPRouter learns a direct feed-forward classifier from query features to model IDs, and MFRouter factorizes the query–model performance matrix to exploit latent compatibility patterns. These methods are attractive because they are computationally lightweight during inference. However, they implicitly assume that proximity in the query's surface-level semantic space is a reliable proxy for proximity in terms of capability demand. In practice, semantically similar queries may require drastically different task difficulty. Consequently, this conflation of surface semantics with capability demand severely degrades routing precision under distribution shifts. Unlike these semantic-only classifiers, we design a dedicated demand-side encoder that avoids ID-based memorization by projecting queries into a continuous latent capability space.

Overhead-Intensive Routing. To achieve high routing precision, recent approaches have increasingly employed complex architectures and heavily parameterized models. For instance, GraphRouter [11] encodes tasks, queries, and candidate LLMs into latent embeddings as distinct nodes within a unified graph, leveraging historical interaction data as edges to frame model selection as an edge-prediction problem. Similarly, CausalLMRouter [24] explicitly employs a heavyweight generative LLM as the router, fine-tuning it to output discrete model choices. However, such sophisticated structural designs and heavyweight routing backbones inevitably introduce substantial operational overhead. In real-world deployments, it is highly unacceptable for a router's inference latency and computational footprint to rival or even exceed those of the downstream target LLMs. In contrast, DART makes routing decisions via an ultra-lightweight inference-time dot-product between the query's latent demand embedding and the candidate's latent capability embedding. DART reduces the inference-time routing decision to a lightweight, parameter-free operation.

Zero-Shot Generalization and Model Onboarding. A critical requirement for modern LLM routers is the ability to seamlessly adapt to previously unseen models. Conventional lightweight routers, such as MLPRouter, KNNRouter, and MFRouter [16, 22], inherently function as closed-set classifiers whose output dimensions are rigidly constrained by the candidate pool observed during training. When a new LLM is introduced, these methods suffer from a severe cold-start problem [5, 21], rendering the router completely blind to the new model's capabilities without extensive retraining. For instance, GraphRouter [11] requires newly collected few-shot interaction data to initialize embeddings for additional LLM nodes. Similarly, while CausalLMRouter [24] can technically accommodate new model names via prompt engineering, it relies solely on internal parametric memory, which is inherently outdated and cannot reflect the actual behavior of the new model. In contrast, DART breaks this barrier through a supply-side capability initialization mechanism. By leveraging static model metadata and family-graph priors, DART seamlessly anchors unseen models in the shared capability space in a zero-shot manner, thereby eliminating the need for retraining or online interaction.

Despite substantial progress in semantic mapping, structural modeling, and generative routing, existing routers fail to satisfy these core criteria simultaneously. To the best of our knowledge, DART is the first to simultaneously achieve high routing precision, minimal operational overhead, and robust zero-shot generalization through a unified dual-side alignment for LLM routing.

3 Problem Formulation

Let $\mathcal{Q}$ denote the query space and $\mathcal{M} = \{m_1, \ldots, m_N\}$ denote the pool of candidate LLMs. For each query $q \in \mathcal{Q}$, an LLM router aims to select an optimal model $m^* \in \mathcal{M}$ that maximizes a routing utility that balances expected performance and inference cost. Conventional query-centric routers formulate model selection as a semantic classification problem, learning a mapping function $f : \mathcal{Q} \to \{1, \ldots, N\}$. However, this rigid ID-based paradigm tends to memorize surface-level semantics and discrete query-model interactions, making zero-shot cold-start onboarding impossible when a new model is added to $\mathcal{M}$.

To overcome the limitations of mapping surface semantics to discrete model IDs, we reformulate LLM routing as a demand-supply alignment problem within a shared low-rank latent capability space $\mathbb{R}^K$. Specifically, each query q is mapped to a latent demand embedding $\mathbf{z}_q \in \mathbb{R}^K$, representing

its directional capability requirements and inherent difficulty, while each candidate model m is associated with a latent capability embedding $\mathbf{u}_m \in \mathbb{R}^K$. The routing objective is thus transformed into selecting the model whose capability best aligns with the query's demand under a cost constraint:

$$m^*(q) = \arg\max_{m \in \mathcal{M}} \big(\langle \mathbf{z}_q, \mathbf{u}_m \rangle - \lambda \cdot cost(m) \big), \tag{1}$$

where $\langle \mathbf{z}_q, \mathbf{u}_m \rangle$ denotes the demand-capability alignment score, $cost(m)$ is the normalized inference cost of model m, and λ is a user-defined cost sensitivity factor. Crucially, this demand-capability alignment does not require online query-model interaction logs, as the supply side can be inferred from readily available offline priors (e.g., public benchmark leaderboards, model metadata, and architectural family graphs). Even when such offline observations are sparse, these structured priors provide sufficient evidence to infer the latent capability $\mathbf{u}_m$. Under this formulation, the core challenges of zero-shot routing are twofold: i) how to effectively derive the supply-side capability embedding $\mathbf{u}_m$ without relying on extensive online interaction logs, and ii) how to distill the demand-side embedding $\mathbf{z}_q$ to align explicitly within the shared capability space.

4 Methodology

4.1 Overview of DART

Figure 2 illustrates an overview of our DART framework. To achieve zero-shot cost-sensitive routing, we systematically decouple DART into three main components: i) supply-side capability modeling, ii) demand-side encoding, and iii) inference-time routing. Specifically, on the supply side, DART constructs a shared latent capability space by factorizing an offline performance matrix derived from offline observations, yielding a latent capability embedding for each candidate model. To seamlessly handle newly introduced LLMs, we introduce a zero-shot initialization mechanism that anchors these new models in the same capability space using family-graph priors and static model metadata. In parallel, given an input query, the demand-side encoder maps it to a latent demand embedding. This encoder is explicitly designed to capture the query's directional capability requirements and inherent difficulty, transcending surface-level semantics. At inference time, the resulting demand and capability embeddings are aligned in the shared latent space. The router then selects the optimal model by combining the demand-capability alignment score with a query-adaptive cost-sensitivity term, accommodating different performance-cost routing scenarios. This design yields a lightweight routing pipeline that avoids the need for extra heavyweight semantic classification models.

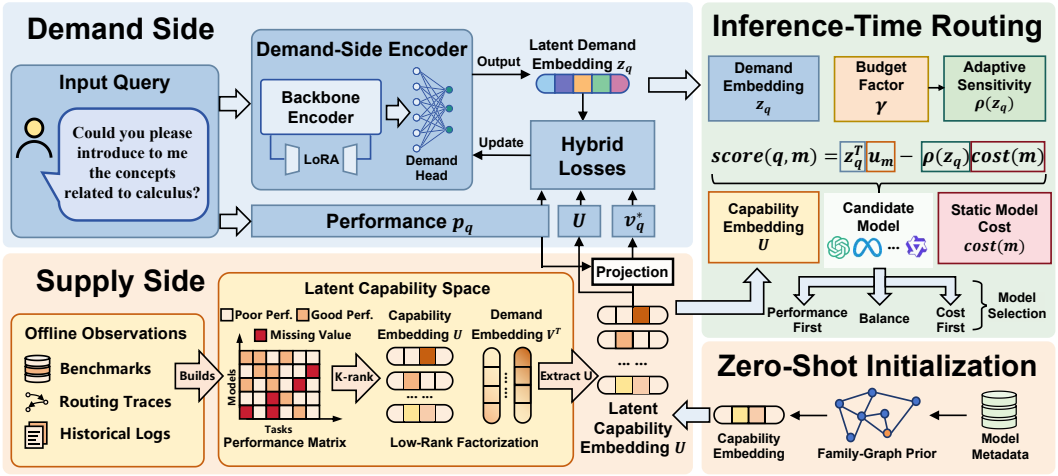


Figure 2: Overview of our DART framework.

4.2 Supply-Side Capability Modeling and Zero-Shot Initialization

The primary objective of the supply side is to assign a capability embedding to each candidate model, establishing a latent capability space for demand-capability alignment.

Capability Modeling via Matrix Factorization. On the supply side, we construct an offline performance matrix $\mathbf{P} \in \mathbb{R}^{N \times B}$, where each row corresponds to a candidate model and each column

to an offline benchmark dimension or task family. Each entry $P_{m,b}$ records a performance observation that can be easily obtained from public sources, such as benchmark leaderboards, routing traces, and historical evaluation logs. We then factorize $\mathbf{P}$ into a rank-K latent capability space:

$$\mathbf{P} \approx \mathbf{U}\mathbf{V}^\top, \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{N\times K}$ comprises the model embeddings and $\mathbf{V} \in \mathbb{R}^{B\times K}$ contains the task requirement factors. The m-th row $\mathbf{u}_m \in \mathbb{R}^K$ of $\mathbf{U}$ serves as the latent capability embedding of model m. Once a factorization is selected, the model capability embeddings and query-level capability targets are defined in the resulting coordinate system, and the demand encoder is trained to predict targets in that same space. Therefore, DART relies on the consistency of the shared latent space rather than on the uniqueness of a particular factorization. The dimensionality K of the latent capability space serves as an information bottleneck, compressing the models' raw performance into shared capability factors (e.g., mathematical reasoning or coding proficiency). Crucially, this low-rank factorization not only extracts these fundamental latent factors but also systematically suppresses noise arising from task-specific evaluation variance and isolated outliers, thereby yielding more robust and generalizable capability representations.

Zero-Shot Initialization via Structured Priors. For any model with complete performance observations in P, we obtain its latent capability embedding via low-rank factorization. However, when a newly introduced model lacks complete performance observations, we initialize its embedding via structured priors, enabling zero-shot onboarding without collecting new query–model interactions or retraining the demand encoder. We first encode its static model metadata, including family, size, context length, and price, into a numeric feature vector, which is then deterministically projected into the latent capability space to form a metadata anchor $\mathbf{a}^{\text{meta}}_{m_{\text{new}}} \in \mathbb{R}^K$. Because this metadata signal alone is weak for capability modeling, we augment it by constructing a family graph over all candidate models. In this graph, the edge weight between two models m_i and m_j is defined as:

$$w_{ij} = \mathbb{I}[f_i = f_j] + \frac{\min(s_i, s_j)}{\max(s_i, s_j)}, \tag{3}$$

where f_i denotes the family of model m_i and s_i denotes its parameter scale. This weighting strategy captures the intuition that models sharing the same lineage and exhibiting similar scale occupy proximal regions in the capability space. Letting $\mathcal{N}(m_{\text{new}})$ denote the top-weighted neighbors of the new model with known embeddings, we aggregate their latent representations into a graph anchor via weighted averaging as:

$$\mathbf{a}^{\text{graph}}_{m_{\text{new}}} = \frac{\sum_{j\in\mathcal{N}(m_{\text{new}})} w_{\text{new},j}\,\mathbf{u}_j}{\sum_{j\in\mathcal{N}(m_{\text{new}})} w_{\text{new},j}}. \tag{4}$$

We then form the zero-shot initialization by fusing the metadata anchor with the graph anchor. If a small subset of benchmark observations subsequently becomes available for the new model, we further refine this prior rather than discarding it. Letting $\Omega(m_{\text{new}}) \subseteq \{1,\dots,B\}$ denote the observed dimensions with corresponding scores $\mathbf{p}_\Omega \in \mathbb{R}^{|\Omega|}$, we compute an observation anchor via least squares over the restricted task basis $\mathbf{V}_\Omega \in \mathbb{R}^{|\Omega|\times K}$ as:

$$\mathbf{a}^{\text{obs}}_{m_{\text{new}}} = \arg\min_{\mathbf{u}\in\mathbb{R}^K} \|\mathbf{p}_\Omega - \mathbf{V}_\Omega\mathbf{u}\|_2^2\,. \tag{5}$$

The final capability embedding $\mathbf{u}_{m_{\text{new}}}$ is formed by a simple addition of all available signals: $\mathbf{u}_{m_{\text{new}}} = \mathbf{a}^{\text{meta}}_{m_{\text{new}}} + \mathbf{a}^{\text{graph}}_{m_{\text{new}}} + \mathbf{a}^{\text{obs}}_{m_{\text{new}}}$. This naturally reverts to the prior form when no observations are available ($\mathbf{a}^{\text{obs}}_{m_{\text{new}}} = \mathbf{0}$). Consequently, our approach treats partial observations merely as an optional correction to the structured priors rather than a strict dependency, thereby shifting the empirical burden of model onboarding from expensive online routing interactions to readily available static metadata and sparse offline measurements.

4.3 Demand-Side Encoding for Latent Demand Embedding

Capability-Space Target Alignment. The demand-side encoder consists of a pretrained backbone adapted with LoRA [13] and a lightweight demand head. Specifically, the backbone extracts contextual query features from surface-level semantics, while fine-tuned LoRA adapters explicitly modulate these features to capture the underlying directional capability requirements and inherent difficulty. The demand head then maps these modulated features into a latent demand embedding $\mathbf{z}_q \in \mathbb{R}^K$. To

explicitly align it with the supply-side representations, we derive a capability-space target $\mathbf{v}_q^\star$ to act as a supervision signal for each query by projecting its observed performance vector $\mathbf{p}_q$ onto the latent space spanned by the known capability embeddings via least squares:

$$\mathbf{v}_q^\star = \arg\min_{\mathbf{v}\in\mathbb{R}^K} \left\|\mathbf{p}_q - \mathbf{U}_{\Omega_q}\mathbf{v}\right\|_2^2, \tag{6}$$

where $\mathbf{U}_{\Omega_q}$ contains the capability embeddings of the models for which the performance of query q is observed. Since $\mathbf{v}_q^\star$ is computed from the same matrix $\mathbf{U}$, it lies in the same latent coordinate system as the model embeddings and provides the supervision signal for demand–capability alignment.

Hybrid Alignment Objectives. To calibrate the geometry, scale, and local structure of the demand embeddings without relying on discrete ID-based classification, we train the demand encoder with three complementary objectives. First, the directional alignment loss $\mathcal{L}_{\text{align}} = 1 - \cos(\mathbf{z}_q, \mathbf{v}_q^\star)$ provides the primary geometric constraint, ensuring $\mathbf{z}_q$ is oriented toward the model capability patterns implied by the observed routing outcomes. Second, the difficulty calibration loss $\mathcal{L}_{\text{diff}} = (h_\phi(\|\mathbf{z}_q\|_2) - d_q)^2$ regresses the norm of the demand embedding to a continuous routing-derived difficulty scalar d_q. Specifically, h_ϕ is a single linear layer that maps the scalar norm $\|\mathbf{z}_q\|_2$ to the predicted difficulty score. d_q is computed as one minus the mean observed performance of query q on the candidate models. This forces harder queries to occupy a different scale regime, effectively making the demand norm a robust geometric proxy for the amount of capability a query requires during cost-sensitive routing. Third, the contrastive structure loss $\mathcal{L}_{\text{con}} = -\frac{1}{N_b}\sum_{i=1}^{N_b}\sum_{j=1}^{N_b} T_{ij}\log Q_{ij}$ aligns the batch-wise topology, where N_b is the batch size. We obtain Q_{ij} from the pairwise similarities of ℓ_2-normalized demand embeddings and T_{ij} from the corresponding projected capability targets. This loss encourages semantically similar queries with different capability demands to separate in the latent space. Finally, these terms are combined into a composite objective:

$$\mathcal{L} = \alpha_{\text{align}}\mathcal{L}_{\text{align}} + \alpha_{\text{diff}}\mathcal{L}_{\text{diff}} + \alpha_{\text{con}}\mathcal{L}_{\text{con}}, \tag{7}$$

where α_{align}, α_{diff}, and α_{con} are balancing weights. To minimize hyperparameter tuning, we simply set all weights to 1.0, granting equal importance to each constraint. Consequently, the objective function remains tightly consistent with DART's core philosophy of deferring routing decisions to the inference-time matching rule rather than hard-coding them during encoder training.

4.4 Inference-Time Routing with Query-Adaptive Cost Sensitivity

At inference time, DART combines the latent demand embedding with the supply-side capability embeddings via a lightweight matching rule modulated by a query-adaptive cost-sensitivity term.

Capability Matching and Cost-Aware Routing. Given a query q, the demand-side encoder produces $\mathbf{z}_q$. Each candidate model $m \in \mathcal{M}$ is represented by its capability embedding $\mathbf{u}_m$ and a normalized static cost $cost(m)$ derived from metadata. The router first computes a demand-capability alignment score $s(q,m) = \mathbf{z}_q^\top \mathbf{u}_m$ via the batched dot product. Before introducing the cost penalty, these raw alignment scores $s(q,m)$ can be normalized across candidates to improve comparability with normalized costs $cost(m)$. To advance the static sensitivity λ defined in Eq. 1 and balance performance and cost, we introduce a query-adaptive cost sensitivity $\rho(\mathbf{z}_q)$ that penalizes expensive models according to the query's inherent difficulty. The final routing decision is then given by:

$$m^*(q) = \arg\max_{m\in\mathcal{M}} \left(s(q,m) - \rho(\mathbf{z}_q)\, cost(m)\right). \tag{8}$$

By explicitly decoupling demand-capability alignment from the cost constraint, this design ensures a single trained router can handle diverse utility regimes by modulating the inference-time tradeoff, without retraining.

Query-Adaptive Cost Sensitivity Formulation. The sensitivity term $\rho(\mathbf{z}_q)$ governs how strongly cost penalizes the alignment score. We consider two implementations: i) a parameter-free analytical form derived from the demand norm, and ii) a learned neural module:

$$\rho_{\text{free}}(\mathbf{z}_q) = \frac{\gamma}{\|\mathbf{z}_q\|_2 + \epsilon}, \qquad \rho_{\text{learned}}(\mathbf{z}_q) = \text{softplus}\left(g_\rho(\mathbf{z}_q)\right). \tag{9}$$

In the parameter-free variant, γ acts as a global budget factor that controls the overall strength of the cost penalty, while ϵ (e.g., 10^{-3}) is a small constant added for numerical stability. This design treats

the norm of the latent demand embedding as a natural proxy for budget preference, thereby making it an ultra-lightweight, entirely parameter-free operation. The learned variant uses a small MLP, $g_\rho(\cdot)$, to capture nonlinear, query-dependent budget preferences, with softplus ensuring non-negativity.

By design, the inference rule remains computationally lightweight. All heavy computation is absorbed into offline capability modeling and demand-side encoding. Online routing merely requires computing one demand embedding, evaluating its alignment with the candidate capability matrix, and subtracting the cost penalty. This preserves DART's core practical advantage, enabling the router to be trained once, extended to unseen models via supply-side zero-shot initialization, and reused across different cost constraints entirely at inference time.

4.5 Implementation of DART

Algorithm 1 summarizes the DART pipeline, which consists of supply-side capability modeling, demand-side encoder training, and inference-time routing. DART first factorizes the offline performance matrix to obtain capability embeddings for candidate models, and then anchors cold-start models by combining metadata and family-graph priors when observed performance is sparse or entirely absent. We train the demand-side encoder by extracting a query embedding, projecting it into a latent demand embedding, deriving an explicit capability-space target, and jointly optimizing the alignment, difficulty calibration, and contrastive losses. This ensures that query embeddings are explicitly mapped into the same capability space as the models. At inference time, the optimized encoder maps each test query to a demand embedding, derives a query-adaptive cost sensitivity from its norm, and computes the final routing score by combining the demand-capability alignment score with a cost penalty, then returns the selected model. By decoupling capability modeling from semantic matching, DART simultaneously achieves highly precise demand-capability alignment, zero-shot cold-start onboarding, and lightweight cost-aware routing within a unified framework.

Algorithm 1 Implementation of DART

Input: i) $\mathcal{Q}_{\text{train}}$, a training query dataset; ii) $\mathcal{M}$, candidate model pool; iii) $\mathbf{P}, \mathbf{E}^{meta}, \mathcal{G}$, offline performance matrix, metadata, and family graph; iv) $cost(\cdot)$, normalized static model costs; v) $g_\theta(\cdot)$, a demand-side encoder parameterized by θ. vi) N_b, E, batch size and epochs; vii) $\alpha_{\text{align}}, \alpha_{\text{diff}}, \alpha_{\text{con}}, \gamma, \epsilon$, hyperparameters.
Output: m^*, optimized routing decision for a query q_{test}.

/ Supply-Side Capability Modeling (Sec. 4.2) */*
$\mathbf{U}, \mathbf{V} \leftarrow \text{Factorize}(\mathbf{P})$
for each $m \in \mathcal{M}$ with sparse observations **do**
  $\mathbf{u}_m \leftarrow \text{CapabilityInit}(\mathbf{e}_m^{meta}, \mathcal{G}, \mathbf{U})$
end for
/ Demand-Side Encoder Training (Sec. 4.3) */*
for $epoch = 1$ to E **do**
  while not finished **do**
    $\mathcal{B} \leftarrow \text{Sample}(\mathcal{Q}_{\text{train}}, N_b)$
    for each $q \in \mathcal{B}$ **do**
      $\mathbf{z}_q \leftarrow g_\theta(q)$
      $\mathbf{v}_q^\star \leftarrow \arg\min_{\mathbf{v}} \|\mathbf{p}_q - \mathbf{U}_{\Omega_q}\mathbf{v}\|_2^2$
      $d_q \leftarrow 1 - \text{mean}(\mathbf{p}_q)$
    end for
    $\mathcal{L} \leftarrow \alpha_{\text{align}}\mathcal{L}_{\text{align}} + \alpha_{\text{diff}}\mathcal{L}_{\text{diff}} + \alpha_{\text{con}}\mathcal{L}_{\text{con}}$
    $\theta \leftarrow \text{update}(\theta, \mathcal{L})$
  end while
end for
/ Inference-Time Routing (Sec. 4.4) */*
Receive a test query q_{test}
$\mathbf{z}_{q_{\text{test}}} \leftarrow g_\theta(q_{\text{test}})$
$\rho \leftarrow \gamma / (\|\mathbf{z}_{q_{\text{test}}}\|_2 + \epsilon)$
$m^* \leftarrow \arg\max_{m \in \mathcal{M}} \left(\mathbf{z}_{q_{\text{test}}}^\top \mathbf{u}_m - \rho \cdot cost(m)\right)$
return m^*

5 Experiments

To evaluate DART's effectiveness and efficiency, we implemented the framework in PyTorch [18]. All experiments were conducted on a workstation equipped with a single NVIDIA RTX 4090 GPU. Please refer to Appendix A for detailed experimental implementations. Due to space limitations, we presented all ablation studies in Appendix C, including detailed investigations on the impact of different losses, latent ranks, and backbone encoders.

5.1 Experimental Setup

Dataset Settings. In the experiments, we investigated seven different benchmarks, involving three commonsense reasoning benchmarks (i.e., HellaSwag [28], Winogrande [20], and ARC Challenge [6]), one knowledge-based language understanding benchmark (i.e., MMLU [12]), one multi-turn conversation benchmark (i.e., MT-Bench [30]), one mathematical problem-solving benchmark (i.e., GSM8K [7]), and one code generation benchmark (i.e., MBPP [3]), totaling 86 tasks. For fair comparisons, we split the training and test sets for these tasks according to RouterBench [14].

Baseline Settings. We compared DART with five representative state-of-the-art (SOTA) routers, i.e., MLPRouter [22], KNNRouter [22], MFRouter [16], GraphRouter [11] and CausalLMRouter [24]. Specifically, MLPRouter, KNNRouter, and MFRouter are lightweight routing models with low overhead. GraphRouter and CausalLMRouter are structurally complex routing paradigms that can achieve satisfactory routing precision. KNNRouter, GraphRouter, and CausalLMRouter can generalize to newly introduced LLMs at inference time. By covering diverse SOTA baselines with high routing precision, low operational overhead, and zero-shot generalization, we can comprehensively evaluate DART's superiority.

Evaluation Metrics. We evaluated the effectiveness of routers based on the outcomes of the models they select, using three metrics: **Performance**, **Cost**, and **Utility**. Specifically, performance indicates the selected model's average response quality, and cost reflects the per-query monetary expense. Utility is the primary deployment-oriented metric that combines performance and cost into a single score: Utility = $\alpha \times Performance - \beta \times Cost$, where $\alpha + \beta = 1$. To systematically evaluate routers under various deployment priorities, following the experimental settings in [11], we computed the utility of three performance-cost routing scenarios: i) **Performance-First** (α = 1.0, β = 0.0, and γ = 0.0), where utility reduces to performance alone; ii) **Balance** (α = 0.5, β = 0.5, and γ = 0.5), emphasizing equitable consideration of both metrics; and iii) **Cost-First** (α = 0.2, β = 0.8, and γ = 0.8), prioritizing economic efficiency while retaining minimal performance sensitivity.

Candidate Model Pool. By default, the candidate model pool includes 10 diverse LLMs covering various parameter scales, price tiers, and families, i.e., WizardLM 13B V1.2 [26], Claude Instant V1, Claude V1, Claude V2 [2], GPT-3.5 Turbo 1106, Code Llama Instruct 34B [19], Llama-2 70B [23], Mistral 7B [15], Mixtral 8x7B [1], and Yi 34B [27]. Meanwhile, to evaluate the router's zero-shot generalization ability, i.e., its ability to accommodate newly introduced models without retraining, we incorporated GPT-4 1106 Preview [17] in the generalization experiment. We also conducted Leave-One-Model-Out (LOMO) experiments over all 11 candidate models.

5.2 Experimental Results

Effectiveness Comparison. Table 1 compares the effectiveness of DART and representative baselines in three performance-cost routing scenarios. From this table, we find that DART achieves the highest utility across all scenarios, significantly outperforming the baselines. Specifically, among the baselines, MLPRouter and KNNRouter achieve moderate utility in the performance-first scenario but degrade significantly under balance and cost-first constraints. This is because their reliance on query-embedding proximity conflates semantic similarity with the capability requirements. In comparison, MFRouter exhibits the weakest overall effectiveness, indicating that matrix factorization without explicit demand-side adaptation fails to capture query-specific directional capability requirements and inherent difficulty. Moreover, although CausalLMRouter leverages a capable language model for routing inference, its prompt-based selection remains tied to surface-level semantics and generation noise. By explicitly encoding query demands and model capabilities within a shared low-rank latent capability space, DART effectively decouples routing from surface semantics, yielding consistent utility improvements over baselines across all scenarios.

Table 1: Effectiveness comparison of methods under different performance-cost routing scenarios.

Method	Performance-First			Balance			Cost-First		
	Perf.	Cost	Utility	Perf.	Cost	Utility	Perf.	Cost	Utility
MLPRouter	0.6847	0.0194	0.6847	0.6111	0.0022	0.3044	0.6381	0.0013	0.1266
KNNRouter	0.6968	0.0067	0.6968	0.5834	0.0046	0.2894	0.5814	0.0040	0.1131
MFRouter	0.6536	0.0023	0.6536	0.5051	0.0018	0.2516	0.4814	0.0015	0.0951
GraphRouter	0.7677	0.0129	0.7677	0.5814	0.0040	0.2887	0.6529	0.0011	0.1297
CausalLMRouter	0.6691	0.0028	0.6691	0.5841	0.0027	0.2907	0.5604	0.0027	0.1099
DART*	0.8265	0.0038	0.8265	0.8258	0.0038	0.4110	0.8265	0.0038	0.1622
DART	0.8636	0.0367	**0.8636**	0.8447	0.0133	**0.4157**	0.8312	0.0024	**0.1643**

Learnable Module Comparison. We considered a variant "DART*" that employs a learnable module to determine the cost sensitivity $\rho(\mathbf{z}_q)$, while DART is parameter-free and derives $\rho(\mathbf{z}_q)$ directly from the norm of the demand embedding. Table 1 shows that variant "DART*" underperforms DART in all scenarios. This is because the learned MLP module $g_\rho(\cdot)$ tends to overfit the specific cost-tradeoff patterns present in the training queries. It fails to generalize robustly to unseen queries with different

difficulty levels. In contrast, parameter-free DART uses the geometric norm of the latent demand embedding, $||z_q||_2$, as an intrinsic structural proxy of the difficulty of the query. This preserves the continuous, globally consistent geometry of the latent capability space without introducing additional parameterized modules that are prone to overfitting.

Table 2: Comparison of time overhead, number of parameters, and end-to-end LLM latency.

Method	Time Overhead (ms)			# of Parameters			LLM Latency (ms)
	Encoder	Routing	Total	Encoder	Routing	Total	
MLPRouter	23.39	0.40	**23.79**	148.66M	107,338	148.77M	1,315.56
KNNRouter	23.47	94.46	117.93	148.66M	**0**	**148.66M**	1,293.51
MFRouter	24.37	0.13	24.50	148.76M	1,408	148.76M	**1,142.43**
GraphRouter	24.52	134.04	158.56	148.66M	58,498	148.71M	1,223.55
CausalLMRouter	**0.02**	121.61	121.63	**0**	3212.57M	3212.57M	1,252.46
DART*	25.08	0.31	25.39	148.86M	49	148.86M	1,215.31
DART	25.12	**0.18**	25.30	148.86M	**0**	148.86M	1,183.03

Routing Overhead Comparison. Table 2 compares time overhead, parameter count, and real end-to-end LLM API latency for each router method. We find that although lightweight routers such as MLPRouter and MFRouter incur extremely low time overhead, their utility remains limited (see Table 1). Because structured methods such as KNNRouter and GraphRouter rely on computationally expensive operations like neighbor search and graph inference, they incur substantially higher routing latency, accounting for about 10% of LLM latency. Furthermore, CausalLMRouter relies on an LLM to select another LLM, adding substantial parameter overhead and causing severe time delays of 121.63 ms. In comparison, DART's encoder merely uses a LoRA branch with a negligible 1.73 ms overhead relative to MLPRouter, while its routing mechanism remains parameter-free (or uses just 49 parameters in DART*). In its default variant, DART's routing head is parameter-free and takes only 0.18 ms, while the demand encoder requires 25.12 ms. This yields a total router latency of 25.30 ms, keeping end-to-end time dominated by downstream LLM inference and demonstrating exceptional deployment efficiency.

Table 3: Utility and growth rate comparison between routing methods introducing an unseen LLM.

Method	Performance-First			Balance			Cost-First		
	Utility	Utility*	Δ	Utility	Utility*	Δ	Utility	Utility*	Δ
KNNRouter	0.6968	0.6968	0.00%	0.2894	0.2894	0.00%	0.1131	0.1131	**0.00%**
GraphRouter	0.7677	0.7481	-2.55%	0.2887	0.2676	-7.31%	0.1297	0.1297	**0.00%**
CausalLMRouter	0.6691	0.6691	0.00%	0.2907	0.2400	-17.44%	0.1099	0.0963	-12.37%
DART	**0.8636**	**0.8685**	**+0.57%**	**0.4157**	**0.4169**	**+0.29%**	**0.1643**	**0.1643**	**0.00%**

Generalization on Unseen LLMs. Table 3 presents the new utility of each routing method (i.e., Utility*) when an unseen LLM (i.e., GPT-4 1106 Preview) is introduced, and their growth rates, i.e., Δ = (Utility*-Utility)/Utility, compared to standard values. We find that DART consistently achieves the highest utility and non-negative growth rates in all three performance-cost routing scenarios. This is because DART leverages model metadata and the family-graph priors to establish demand-capability alignment without retraining or few-shot optimizations. In contrast, KNNRouter operates as a closed-set selector and cannot adapt to newly added models, so its utility remains unchanged. CausalLMRouter lacks explicit structural priors for unseen models, leading to substantial utility drops of 17.44% and 12.37% in the balance and cost-first scenarios, respectively. Furthermore, we investigated DART's generalization ability when the performance matrix for the unseen model is incomplete (i.e., only a portion of the performance values is available). Figure 3 illustrates the utility of DART with different availability ratios of the performance matrix. We find that even when only 0% or 20% of values in the matrix are available, DART still achieves strong utility by anchoring model capability embeddings through metadata and family-graph priors, highlighting the superior generalization ability of our approach.

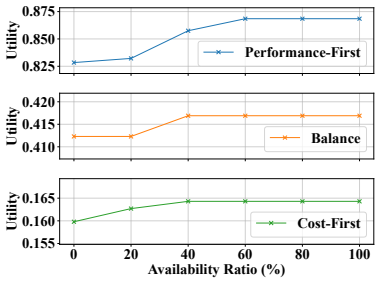


Figure 3: DART's utility with different availability ratios.

6 Conclusion

In this paper, we introduced DART, a novel routing framework grounded in demand-supply alignment to optimize LLM performance-cost tradeoffs. By decoupling the routing process into a shared low-rank latent capability space, DART shifts the paradigm away from superficial semantic matching. By integrating an offline performance matrix, zero-shot capability initialization, distilled demand-capability alignment, and query-adaptive cost sensitivity, DART successfully satisfies the three core criteria of an ideal router. Extensive experiments across comprehensive benchmarks demonstrate that DART achieves state-of-the-art utility under diverse cost constraints, while detailed ablation studies confirm the indispensable role of each architectural component. Despite these advancements, a primary limitation of this paper is our reliance on static cost optimization (i.e., public token pricing) without accounting for unpredictable token-generation volumes or real-time inference latency. Therefore, future work will focus on integrating dynamic, system-level feedback metrics to further refine routing precision, execution efficiency, and generalization in live production deployments.

Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 62272170).

References

[1] Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[2] Anthropic. Model Card and Evaluations for Claude Models. `https://www-cdn.anthropic.com/bd2a28d2535bfb0494cc8e2a3bf135d2e7523226/Model-Card-Claude-2.pdf`, 2023. Accessed: Mar. 23, 2026.

[3] Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[4] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[5] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowl. Based Syst.*, 46:109–132, 2013.

[6] 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.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4171–4186, 2019.

[9] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: cost-efficient and quality-aware query routing. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[10] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[11] Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for LLM selections. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[13] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[14] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-LLM routing system. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[15] Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[16] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[17] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

[19] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

[20] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 8732–8740, 2020.

[21] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, pages 253–260, 2002.

[22] Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. Tensoropera router: A multi-model router for efficient LLM inference. In Franck Dernoncourt, Daniel Preotiuc-Pietro, and Anastasia Shimorina, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 452–462, 2024.

[23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut

Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[24] Asterios Tsiourvas, Wei Sun, and Georgia Perakis. Causal LLM routing: End-to-end regret minimization from observational data. *arXiv preprint arXiv:2505.16037*, 2025.

[25] Per-Åke Wedin. Perturbation bounds in connection with singular value decomposition. *BIT Numerical Mathematics*, 12(1):99–111, 1972.

[26] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[27] Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. Yi: Open foundation models by 01.ai. *arXiv preprint arXiv:2403.04652*, 2024.

[28] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 4791–4800, 2019.

[29] Tuo Zhang, Asal Mehradfar, Dimitrios Dimitriadis, and Salman Avestimehr. Leveraging uncertainty estimation for efficient LLM routing. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

[30] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

A Detailed Experimental Setup

A.1 Implementation Details

Performance Matrix Construction. In our experiments, we constructed the offline performance matrix $\mathbf{P}$ from the RouterBench training split. Specifically, we grouped routing records by model and task source and averaged duplicate model–source entries to obtain a model–task performance matrix. We subtracted the global mean from the imputed matrix and computed a deterministic reduced Singular Value Decomposition (SVD). Retaining the leading K singular components yields the rank-K latent factorization. We used the resulting factor $\mathbf{U}$ both as the model capability embedding matrix and to project query-level performance vectors into the latent basis. We did not use the test split for matrix construction, factorization, or demand-side encoder training.

Zero-Shot Initialization via Structured Priors. Conventional routing methods struggle with the cold-start problem because they lack a foundational understanding of the newly introduced models. Consequently, true zero-shot routing is hard to attain, necessitating extensive retraining or expensive online few-shot interactions. To address this, DART introduces a supply-side capability initialization mechanism that enables seamless zero-shot onboarding. Specifically, we first extracted model metadata from the candidate pool to construct static model features, including model family, parameter size, context length, and input/output price. Because this metadata signal alone is inherently weak for capability modeling, we augmented it by constructing a family graph over all candidate models. When performance observations are entirely absent, we used the resulting family-graph

priors to robustly anchor the new model within the shared latent capability space. Furthermore, even with only sparse performance observations, we can still leverage the family graph to aggregate a more robust capability embedding.

Zero-Shot Onboarding Protocol. For each LOMO trial, we excluded the held-out model from all observations used to construct the capability space and train the demand encoder. At test time, DART initializes the held-out model's capability embedding using only metadata and family-graph priors, and evaluates routing over the expanded pool without retraining. We compared this zero-shot setting with a full-observation reference in which the held-out model's training-split benchmark observations are available. The full-observation setting is not a deployable zero-shot method and serves only as a reference for measuring the remaining initialization gap.

A.2 Baseline Settings

We evaluated DART against five representative state-of-the-art routers: MLPRouter [22], KNNRouter [22], MFRouter [16], GraphRouter [11], and CausalLMRouter [24]. Except for CausalLMRouter, which utilizes Llama-3.2-3B-Instruct as its routing backbone, all other baselines and DART variants employ Longformer [4] as their default query encoder. Existing baselines inherently treat routing as a classification problem, mapping query semantics to discrete model IDs. Their training is based on ground-truth identifications derived by maximizing Utility $= \alpha \times$ Performance $- \beta \times$ Cost. Therefore, adapting to different performance-cost routing scenarios (i.e., varying α and β) necessitates retraining the entire router for each specific constraint. In contrast, DART avoids discrete ID targets entirely, optimizing the encoder towards an explicit capability-space target $\mathbf{v}_q^\star$ (Eq. 6). This decoupled design allows DART to seamlessly handle diverse cost constraints during inference without retraining. To ensure a fair evaluation, we reported the final Utility scores of all methods across the established routing scenarios.

MLPRouter & MFRouter. MLPRouter learns a direct mapping from query features to model IDs, while MFRouter relies on matrix factorization to recover latent compatibility. Because both methods are strictly closed-set models with fixed output dimensions that correspond only to the training candidate pool, they cannot accommodate unseen models. Consequently, we excluded both from the zero-shot generalization experiments.

KNNRouter. KNNRouter relies on nearest-neighbor retrieval for routing. Although it is fundamentally a closed-set method constrained by historical model choices, we specifically retained it in the generalization experiments as a representative closed-set baseline to highlight the behavioral limitations of rigid selection mechanisms when faced with new models.

GraphRouter. GraphRouter propagates signals across a query-model graph. For the zero-shot onboarding experiments, it can technically introduce unseen models by adding new graph nodes. However, without online few-shot interaction logs, it initializes latent embeddings for these newly introduced models randomly.

CausalLMRouter. CausalLMRouter leverages a generative LLM (specifically, Llama-3.2-3B-Instruct) to infer routing decisions. It readily accommodates newly introduced models by simply appending their names to the input prompt, allowing it to participate in the generalization experiments. However, because its architecture inherently relies on a monolithic generative LLM rather than a modular representation encoder, it cannot participate in the backbone encoder ablation experiments and remains fixed during those evaluations.

DART* and DART. We used **DART*** to denote the variant employing a learned cost-sensitivity head. Conversely, **DART** refers to the parameter-free variant, where query-adaptive cost sensitivity is analytically derived directly from the norm of the latent demand embedding. Regardless of the cost-penalty mechanism, both variants share the same underlying latent capability space.

B Additional Experimental Results and Analysis

End-to-End Leave-One-Model-Out Evaluation. To evaluate DART's zero-shot onboarding ability beyond a single illustrative model, we conducted an end-to-end LOMO evaluation over all 11 candidate models. In each trial, one model was treated as a newly introduced unseen LLM. We removed its benchmark observations from the model-task performance matrix and initialized its

capability embedding using only static metadata and family-graph priors, without retraining the demand-side encoder. We then evaluated the complete routing system over the expanded candidate pool. All utilities in this experiment are reported under the Balance setting. From Table 4, DART achieves the highest average end-to-end utility among the compared methods, and outperforms the listed baselines on average for every held-out model.

Table 4: End-to-end utility under leave-one-model-out onboarding. Utilities are reported under the Balance setting.

Held-out Model	**KNNRouter**	**GraphRouter**	**CausalLMRouter**	**DART**
WizardLM-13B-V1.2	0.287	0.291	0.244	0.448
Claude-Instant-V1	0.289	0.233	0.238	0.421
Claude-V1	0.287	0.285	0.238	0.455
Claude-V2	0.289	0.281	0.241	0.428
GPT-3.5-Turbo-1106	0.289	0.258	0.237	0.467
Code-Llama-34B-Instruct	0.289	0.236	0.243	0.441
Llama-2-70B	0.289	0.281	0.238	0.404
Mistral-7B	0.289	0.283	0.326	0.398
Mixtral-8x7B	0.290	0.279	0.239	0.435
Yi-34B	0.285	0.310	0.238	0.471
GPT-4-1106-Preview	0.289	0.268	0.240	0.417
Average	0.288	0.273	0.247	0.435

Task-Space Expansion and Domain-Shift Analysis. To evaluate the stability of the shared low-rank capability space under new task dimensions, we expanded the performance matrix by holding out 10%, 20%, or 30% of task columns at random, or all columns from one domain (mathematics, code, or dialogue). Using the remaining columns, we constructed the original rank-K factorization and trained the demand-side encoder, then compared a frozen-space update, which preserves the original embeddings, encoder, and routing rule, with a retrained update, which refactorizes the expanded matrix and recalibrates the encoder. We averaged random-expansion results over five task splits, and we excluded reporting outcomes from factorization, model selection, and encoder training. Table 5 reports the residual and the cosine of the embedding. The former is the portion of an added task's cross-model performance pattern unexplained by the original latent space, and the latter measures the similarity of the original and updated model embeddings after Procrustes alignment. The table also shows agreement, the fraction of retained-task routing decisions that remain unchanged, and $\Delta\mathrm{Utility}_{\mathrm{retained}}$, the change in retained-task utility relative to the original router. The embedding cosine exceeds 0.97 for random expansions and ranges from 0.979 to 0.999 for domain holdouts, while retained-task utility changes remain within $[-0.005, 0.002]$. Agreement is also generally high, though variation across random expansions shows that expansion size alone does not determine routing stability. Domain holdouts yield residuals between 0.112 and 0.269, suggesting that the original capability space captures substantial cross-domain structure, although the benefit of refactorization depends on the domain. These findings align with low-rank approximation and subspace perturbation theory: a small residual suggests an added task lies close to the existing capability span [10], while stability is expected when the perturbation is small relative to the spectral gap [25]. Accordingly, residual and spectral gap serve as practical diagnostics, and large residuals may signal new capability directions and the need for refactorization and demand-side recalibration.

C Ablation Studies

Impact of Different Losses. To assess the contribution of each loss term, we conducted ablation studies with three variants of DART: "w/o Align", "w/o Diff", and "w/o Contrast", corresponding to removing the respective terms in Equation 7. Table 6 reports the mean $\pm$ standard deviation of utility across five random seeds under three routing scenarios. From the table, the "w/o Align" variant causes the largest utility degradation, particularly in the performance-first and cost-first scenarios. This is because the demand-side encoder no longer receives a direct directional constraint toward the capability-space target, thereby losing the primary signal that aligns demand embeddings with capability embeddings in the shared latent capability space. In the balance scenario, the variant obtains

Table 5: Stability under task-space expansion and whole-domain holdouts.

Expansion	Residual	Emb. cosine	Agreement	Frozen util.	Retrained util.	ΔUtility
Random 10%	0.27 ± 0.14	0.99 ± 0.01	0.84 ± 0.12	0.17 ± 0.11	0.26 ± 0.05	$+0.002 \pm 0.008$
Random 20%	0.31 ± 0.05	0.98 ± 0.01	0.79 ± 0.09	0.17 ± 0.13	0.29 ± 0.04	-0.001 ± 0.003
Random 30%	0.21 ± 0.11	0.98 ± 0.02	0.90 ± 0.18	0.23 ± 0.08	0.26 ± 0.08	0.000 ± 0.003
Leave-Math-Out	0.13	1.00	0.88	0.25	0.26	-0.005
Leave-Code-Out	0.27	0.98	0.86	0.21	0.23	0.000
Leave-Dialogue-Out	0.11	0.99	0.97	0.26	0.26	-0.001

Table 6: Ablation study results on different losses.

Method	Performance-First	Balance	Cost-First	Overall Mean
w/o Align	0.6280 ± 0.0347	$\mathbf{0.4196 \pm 0.0182}$	0.1149 ± 0.0077	0.3891
w/o Diff	0.8529 ± 0.0054	0.4147 ± 0.0019	0.1559 ± 0.0024	0.4745
w/o Contrast	0.8533 ± 0.0020	0.4107 ± 0.0011	0.1508 ± 0.0050	0.4716
Full (DART)	$\mathbf{0.8640 \pm 0.0033}$	0.4155 ± 0.0029	$\mathbf{0.1649 \pm 0.0041}$	**0.4815**

a slightly higher mean utility than the full model. However, this result comes with a substantially larger standard deviation and does not indicate a consistent advantage across random seeds. For variant "w/o Diff", the overall mean utility decreases after removing difficulty calibration, suggesting that query-difficulty calibration is beneficial for cost-sensitive routing. Moreover, variant "w/o Contrast" also exhibits a lower overall mean utility, indicating that contrastive regularization helps preserve routing-relevant relational structure among queries. The experimental results show that demand-capability alignment, difficulty calibration, and contrastive regularization play complementary roles in achieving a stable performance–cost tradeoff across routing scenarios.

Table 7: Ablation study results on latent rank.

Rank	Performance-First			Balance			Cost-First		
	Perf.	Cost	Utility	Perf.	Cost	Utility	Perf.	Cost	Utility
1	0.5591	0.0029	0.5591	0.4767	0.0006	0.2380	0.8285	0.0024	0.1638
2	0.8305	0.0030	0.8305	0.8197	0.0031	0.4083	0.8319	0.0055	0.1620
4	0.8373	0.0161	0.8373	0.8244	0.0036	0.4104	0.8258	0.0035	0.1624
6	0.8636	0.0367	**0.8636**	0.8447	0.0133	**0.4157**	0.8312	0.0024	**0.1643**
8	0.8292	0.0249	0.8292	0.8393	0.0193	0.4100	0.8130	0.0042	0.1592
10	0.8251	0.0194	0.8251	0.8251	0.0079	0.4086	0.8238	0.0095	0.1572

Impact of Latent Rank. To examine how the rank dimension in the low-rank factorization governs the capacity and stability of the latent capability space, Table 7 compares DART performance across different latent ranks. The table shows a pronounced non-monotonic trend across all scenarios, indicating that rank serves as a capacity bottleneck rather than merely a regularization parameter. Specifically, when rank is 1, the capability space is overly compressed, leading to severe underfitting in the performance-first and balance scenarios. This is because the encoder fails to preserve sufficient structure to distinguish models with divergent routing behaviors. Moreover, increasing the rank to 6 yields substantial gains, as the factorization becomes sufficiently rich to capture shared capability factors without absorbing excessive noise. However, higher-dimensional embeddings beyond rank 6 retain more noise, diluting the shared model capability and degrading utility. Note that the cost-first scenario shows relatively little sensitivity to rank variations, suggesting that budget-sensitive routing does not rely heavily on fine-grained distinctions in capabilities. The ablation results show that an intermediate rank strikes an optimal balance, ensuring stable query projection and robustness.

Impact of Backbone Encoders. To evaluate the impact of different backbone encoders, Table 8 presents routing results when all methods employ a BERT [8] backbone instead of a Longformer backbone. From the table, DART retains the highest performance and utility across all three scenarios. This is because the substantial margin comes from jointly optimizing alignment, difficulty calibration, and contrastive regularization within the latent capability space, rather than from backbone capacity

Table 8: Ablation study results on backbone encoders.

Method	Performance-First			Balance			Cost-First		
	Perf.	**Cost**	**Utility**	**Perf.**	**Cost**	**Utility**	**Perf.**	**Cost**	**Utility**
MLPRouter	0.6745	0.0031	0.6745	0.6259	0.0020	0.3120	0.5976	0.0013	0.1184
KNNRouter	0.6853	0.0104	0.6853	0.5766	0.0075	0.2846	0.5773	0.0018	0.1140
MFRouter	0.6536	0.0027	0.6536	0.4801	0.0016	0.2392	0.4895	0.0014	0.0968
GraphRouter	0.7286	0.0255	0.7286	0.6259	0.0029	0.3115	0.7232	0.0040	0.1414
DART	0.8393	0.0110	**0.8393**	0.8244	0.0042	**0.4101**	0.8190	0.0037	**0.1608**

alone. The overall trends of baselines remain consistent with those observed using the Longformer backbone in Table 1. The experimental results confirm that our approach's advantage is not contingent on a specific encoder architecture, demonstrating DART's strong generalization ability.

D Broader Impacts

The DART framework has a profound positive impact on advancing "Green AI" and sustainable computing. As the parameter scale of large language models grows exponentially, the inference process consumes massive computational resources and generates a significant carbon footprint. By precisely aligning user demands with candidate model capabilities within a shared latent space, DART dynamically assigns relatively simple tasks to the most cost-effective, lightweight models, thereby preventing the unnecessary overuse of computation-heavy models. This intelligent matching mechanism significantly reduces the overall computational overhead and energy consumption of LLM systems, providing a practical technical pathway to build environmentally friendly next-generation AI infrastructure.

DART significantly promotes the democratization and accessibility of advanced AI technologies. In the current LLM ecosystem, exorbitant inference costs and complex deployment logic for routing systems often act as formidable barriers for small enterprises, independent developers, and academic institutions. DART provides a zero-shot routing mechanism, enabling users to achieve highly efficient performance-cost routing without collecting massive interaction logs or incurring retraining costs. This substantially lowers the economic and technical thresholds for leveraging cutting-edge AI, empowering innovators with limited budgets to benefit equally from multi-model ecosystems and ultimately stimulating broader, society-wide innovation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper focuses on the application of large language models.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper fully discloses all the information needed to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The paper provides code in the supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper provides the experimental settings in Section 5.1 and Section A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 5 reports mean ± standard deviation over five random task splits for the random task-space expansion settings, while Table 6 reports mean ± standard deviation over five random seeds. The standard deviations reflect variability due to task-split sampling and stochastic training, respectively. The domain holdout results in Table 5 use one predefined split per domain and are reported without error estimates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides information about computer resources in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the potential societal impacts of the work performed in Section D.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper uses popular datasets and models. We cite the original paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.