Logic Distillation: Learning from Code Function by Function for Decision-making Tasks

Dong Chen[1,2,3], **Shilin Zhang**[1], **Fei Gao**[1], **Yueting Zhuang**[4], **Siliang Tang**[4], **Qidong Liu**[1,2,3] *, **Mingliang Xu**[1,2,3] †
[1]The School of Computer and Artificial Intelligence of Zhengzhou University
[2]Engineering Research Center of Intelligent Swarm Systems, Ministry of Education
[3]National Supercomputing Center In Zhengzhou
[4]Zhejiang University
{chendongai,ieqdliu,iexumingliang}@zzu.edu.cn, {iszhangshilin1024,gaofei0191}@gs.zzu.edu.cn,
{yzhuang,siliang}@zju.edu.cn

Abstract

Large language models (LLMs) have garnered increasing attention owing to their powerful comprehension and generation capabilities. Generally, larger LLMs (L-LLMs) that require paid interfaces exhibit significantly superior performance compared to smaller LLMs (S-LLMs) that can be deployed on a variety of devices. Knowledge distillation (KD) aims to empower S-LLMs with the capabilities of L-LLMs, while S-LLMs merely mimic the outputs of L-LLMs, failing to get the powerful decision-making capability for new situations. Consequently, S-LLMs are helpless when it comes to continuous decision-making tasks that require logical reasoning. To tackle the identified challenges, we propose a novel framework called Logic Distillation (LD). Initially, LD employs L-LLMs to instantiate complex instructions into discrete functions and illustrates their usage to establish a function base. Subsequently, LD fine-tunes S-LLMs based on the function base to learn the logic employed by L-LLMs in decision-making. During testing, S-LLMs will yield decision-making outcomes, function by function, based on current states. Experiments demonstrate that with the assistance of LD, S-LLMs can achieve outstanding results in continuous decision-making tasks, comparable to, or even surpassing, those of L-LLMs. The code and data for the proposed method are provided for research purposes https://github.com/Anfeather/Logic-Distillation.

1 Introduction

Large language models (LLMs) [Ouyang *et al.*, 2022], such as GPT-4 [Achiam *et al.*, 2023] and GLM-4 [GLM *et al.*, 2024], have been extensively applied owing to their powerful capabilities like comprehension and generation. Particularly, LLMs demonstrate superior performance in autonomous embodied agents, showcasing advanced decision-making capabilities grounded in comprehending instructions and logical reasoning [Xi *et al.*, 2023].

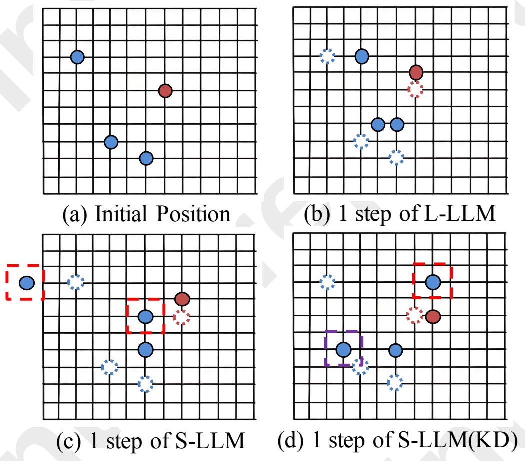


Figure 1: The outcome of one step in the pursuit game.

Despite the remarkable capabilities of LLMs, such as GPT-4 and GLM-4, their substantial computational requirements render them impractical for deployment on most devices [Chen *et al.*, 2024b; Chen *et al.*, 2024a; Chen *et al.*, 2025]. On the other hand, numerous companies have attempted to develop relatively smaller open-source LLMs, including GLM-4-9B [GLM *et al.*, 2024] and LLaMA-7B [Touvron *et al.*, 2023], which are compatible with consumer-grade GPUs like RTX 3090 Ti. In this paper, we refer to LLMs that cannot be deployed on most devices and require invocation through a paid interface as larger LLMs (L-LLMs), in contrast to smaller LLMs (S-LLMs) deployable on consumer-grade GPUs. Generally, L-LLMs exhibit significantly superior performance across various domains, particularly in logical reasoning. Nonetheless, S-LLMs have garnered extensive attention owing to their convenient deployment and cost-free nature. Consequently, an increasing number of researchers are focusing on Knowledge Distillation (KD) of LLMs [Gou

*Corresponding Author
†Corresponding Author

et al., 2021], where L-LLMs act as teachers imparting knowledge, while S-LLMs serve as students, mimicking the outputs of teachers [Xu *et al.*, 2024].

While KD has been demonstrated to effectively enhance the capabilities of S-LLMs in numerous tasks [Dai *et al.*, 2023], endowing S-LLMs with the decision-making capability of L-LLMs continues to pose a significant challenge. As shown in Figure 1, we present the decision-making outcome of a single step in a pursuit game, where LLMs control three blue dots chase an orange dot. Figure 1(a) displays the initial positions of four dots. Meanwhile, in Figure 1(b), the L-LLM (GLM-4) effectively comprehends the game rules, enabling informed decision-making based on the position of the orange dot. In contrast, Figure 1(c) reveals that the S-LLM (GLM-4-9B) lacks proficiency in following rules as the points enclosed by the red dashed line violate the rules of movement. Additionally, Figure 1(d) highlights the issue with KD, where the student merely mimics the output of the teacher without comprehending the logic behind the teacher's decision-making. Specifically, in Figure 1(d), the orange dot moved one unit to the right, while the point enclosed by the purple dashed line moved one unit to the left. Such results stem from the KD process, where the S-LLM remembers the outputs the L-LLM has had in the past. Based on the aforementioned analysis, we summarize the limitations of S-LLMs and KD in decision-making tasks as follows: 1. Ineffectiveness in following complex instructions: Despite extensive fine-tuning, S-LLMs continue to struggle with following intricate rules in decision-making tasks. 2. Failure to comprehend the logic of L-LLMs: Fine-tuned S-LLMs merely mimic the outputs of L-LLMs and lose their decision-making capabilities when encountering unknown scenarios.

Recently, there has been considerable work aimed at establishing the translation between natural language and code, where a sentence is a logical code line [Feng *et al.*, 2020; Chen *et al.*, 2021]. Drawing inspiration from this, we suggest breaking down the decision-making logic of L-LLMs into multiple stages, with each stage being represented by a specific code function. Subsequently, S-LLMs engage in decision-making by learning and applying the pertinent functions. More specifically, we propose Logic Distillation (LD). First, we leverage the powerful comprehension and logical reasoning capabilities of L-LLMs to decompose the rules governing decision-making tasks into multiple stages, forming different functions. Subsequently, we integrate these functions along with their comments, usage examples, etc., into a function base, and use it to fine-tune S-LLMs to enhance their ability to invoke relevant functions. Besides, to alleviate the issue of S-LLMs struggling to follow complex instructions, we propose transforming generation into selection. During each stage of decision-making, S-LLMs will select and invoke functions based on the current state.

As depicted in Figure 2, both KD and LD require guidance from a teacher with superior capabilities. KD emphasizes the imparting of content, which involves students mimicking the teacher's output. In contrast, the proposed LD concentrates on the underlying logic of task execution. The teacher decomposes a complex task into multiple basic logics, represented by functions, enabling the student to make decision function by function.

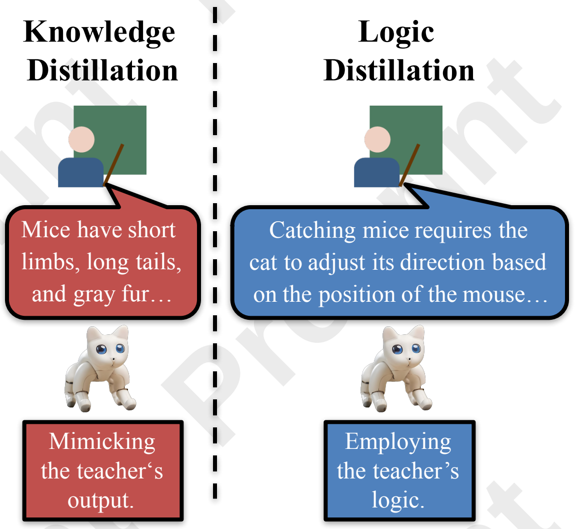

Knowledge Distillation
Logic Distillation
Mice have short limbs, long tails, and gray fur…
Catching mice requires the cat to adjust its direction based on the position of the mouse…
Mimicking the teacher's output.
Employing the teacher's logic.

Figure 2: KD vs LD. KD aims to have smaller models mimic the output of larger models, while LD tries to enable smaller models to understand how larger models accomplish a task.

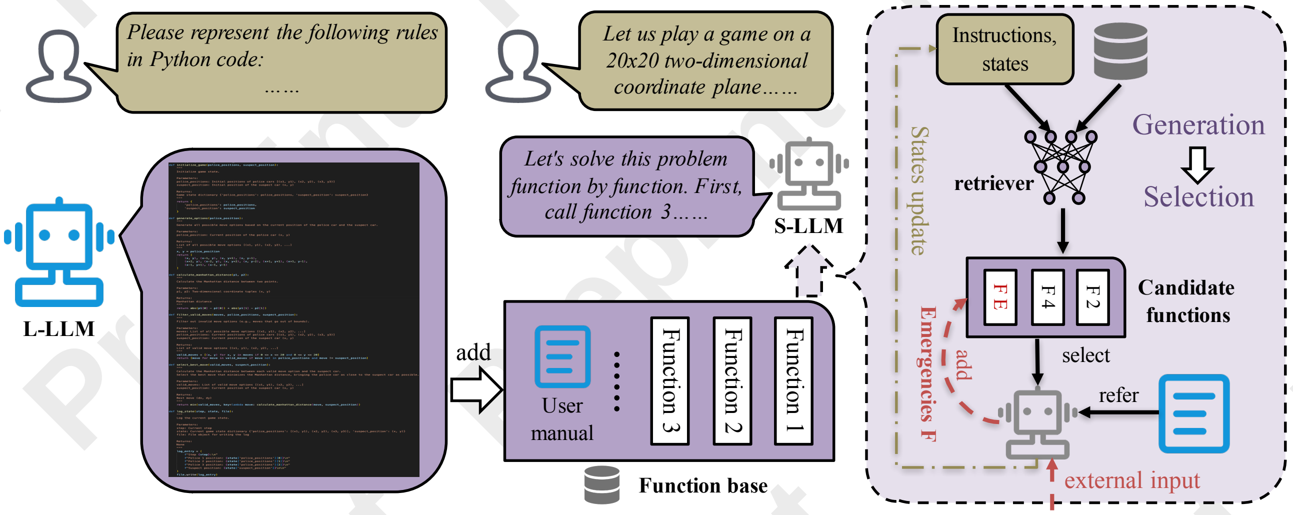
The main contributions of this paper can be summarized as follows:

- We analyzed the issues of Knowledge Distillation in the context of decision-making in the LLMs era.
- We propose Logic Distillation to enable S-LLMs to accomplish decision-making tasks akin to L-LLMs.
- We conducted experiments in different scenarios to validate the effectiveness of the proposed method.

2 Related Work

Large language models (LLMs) [Ouyang *et al.*, 2022; Chowdhery *et al.*, 2022; Thoppilan *et al.*, 2022; Zhao *et al.*, 2023; Koundinya Gundavarapu *et al.*, 2024] are trained on broad data and can be easily adapted to a wide range of tasks [Bommasani *et al.*, 2021], which have been applied to education [Biswas, 2023; Kasneci *et al.*, 2023], healthcare [Thirunavukarasu *et al.*, 2023; Peng *et al.*, 2023; Guo *et al.*, 2024], finance [Wu *et al.*, 2023], etc. However, LLMs with impressive capabilities often suffer from size limitations, making them impractical to run on most devices and costly for invoking their interfaces [Chen *et al.*, 2024b].

In order to endow S-LLMs with the superior capabilities of L-LLMs, Knowledge Distillation (KD) of LLMs has become a focal point of related research [Xu *et al.*, 2024]. In this context, L-LLMs like GPT-4 or GLM-4 are highly skilled teachers, while S-LLMs are students that learn to mimic the outputs of teachers [He *et al.*, 2023; Gu *et al.*, 2024; Agarwal *et al.*, 2024; Liu *et al.*, 2023]. Besides, numerous works have also focused on the reasoning steps of LLMs. Distilling step-by-step [Hsieh *et al.*, 2023] extracts L-LLM rationales as additional supervision for training S-LLMs within a multi-task framework. The Orca framework [Mukherjee *et al.*, 2023] augments the prompt-response data pairs by incorporating a



```python
def initialize_game(police_positions, suspect_position):
    """
    Initialize game state.

    Parameters:
    police_positions: Initial positions of police cars [(x1, y1), (x2, y2), (x3, y3)]
    suspect_position: Initial position of the suspect car (x, y)

    Returns:
    Game state dictionary {'police_positions': police_positions, 'suspect_position': suspect_position}
    """
    return {
        'police_positions': police_positions,
        'suspect_position': suspect_position
    }

def generate_options(police_position):
    """
    Generate all possible move options based on the current position of the police car and the suspect car.

    Parameters:
    police_position: Current position of the police car (x, y)

    Returns:
    List of all possible move options [(x1, y1), (x2, y2), ...]
    """
    x, y = police_position
    return [
        (x, y), (x-1, y), (x, y+1), (x, y-1),
        (x+2, y), (x-2, y), (x, y+2), (x, y-2), (x+1, y+1), (x+1, y-1),
        (x-1, y+1), (x-1, y-1)
    ]

def calculate_manhattan_distance(p1, p2):
    """
    Calculate the Manhattan distance between two points.

    Parameters:
    p1, p2: Two-dimensional coordinate tuples (x, y)

    Returns:
    Manhattan distance
    """
    return abs(p1[0] - p2[0]) + abs(p1[1] - p2[1])

def filter_valid_moves(moves, police_positions, suspect_position):
    """
    Filter out invalid move options (e.g., moves that go out of bounds).

    Parameters:
    moves: List of all possible move options [(x1, y1), (x2, y2), ...]
    police_positions: Current positions of police cars [(x1, y1), (x2, y2), (x3, y3)]
    suspect_position: Current position of the suspect car (x, y)

    Returns:
    List of valid move options [(x1, y1), (x2, y2), ...]
    """
    valid_moves = [(x, y) for x, y in moves if 0 <= x <= 20 and 0 <= y <= 20]
    return [move for move in valid_moves if move not in police_positions and move != suspect_position]

def select_best_move(valid_moves, suspect_position):
    """
    Calculate the Manhattan distance between each valid move option and the suspect car.
    Select the best move that minimizes the Manhattan distance, bringing the police car as close to the suspect car as possible.

    Parameters:
    valid_moves: List of valid move options [(x1, y1), (x2, y2), ...]
    suspect_position: Current position of the suspect car (x, y)

    Returns:
    Best move (dx, dy)
    """
    return min(valid_moves, key=lambda move: calculate_manhattan_distance(move, suspect_position))

def log_state(step, state, file):
    """
    Log the current game state.

    Parameters:
    step: Current step
    state: Current game state dictionary {'police_positions': [(x1, y1), (x2, y2), (x3, y3)], 'suspect_position': (x, y)}
    file: File object for writing the log

    Returns:
    None
    """
    log_entry = (
        f"Step {step}:\n"
        f"Police 1 position: {state['police_positions'][0]}\n"
        f"Police 2 position: {state['police_positions'][1]}\n"
        f"Police 3 position: {state['police_positions'][2]}\n"
        f"Suspect position: {state['suspect_position']}\n\n"
    )
    file.write(log_entry)
```

Figure 3: Illustration of the proposed Logic Distillation (LD). LD consists of three components: L-LLMs, retriever, and S-LLMs. L-LLMs are responsible for decomposing human-provided rules and instantiating them as basic functions to construct a function base. Besides, L-LLMs offer a user manual that explains the usage of these functions, including details such as rule descriptions and code comments. The retriever is in charge of retrieving the top-K functions based on the insturctions and states. S-LLMs select the appropriate functions for different stages of the task. Subsequently, S-LLMs will systematically make decisions function by function.

system message designed to facilitate student models' comprehension of the reasoning process. Subsequently, Orca 2 [Mitra *et al.*, 2023] advances this approach by training the student model to discern the most efficacious solution strategy for individual tasks, guided by the performance metrics established by Orca. However, these methods are essentially still making S-LLMs mimic the outputs of L-LLMs, rather than comprehending the reasoning logic.

3 Methodology

This paper investigates interactive decision-making tasks, where interactions can be divided into multiple steps (each interaction counts as one step), and each step can be further divided into multiple stages (several stages collectively complete one decision-making process).

As illustrated in Figure 3, we explore the Logic Distillation (LD) to empower S-LLMs to engage in decision-making akin to L-LLMs. LD first decomposes rules (instructions) x into multiple stages with L-LLMs and converts the decision-making logic into the corresponding code functions (such as calculating the distance between points). Then, LD will construct a function base containing functions and user manual of these functions. Besides, LD employs a retriever to find the top-K functions that most relevant to the current states and x. Subsequently, S-LLMs will select functions f one by one to make decisions. Overall, LD consists of three components: (i) L-LLMs, (ii) retriever, (iii) S-LLMs.

3.1 L-LLMs

L-LLMs exhibit exceptional capabilities in decision-making. To distill these capabilities into the S-LLMs, we propose instantiating the logic of L-LLMs through functions:

$$p_{\theta_L}(f, u|x) = \prod_i p_{\theta_L}(y_i|x, y_{1:i-1}) \tag{1}$$

where L-LLMs p_{θ_L} parametrized by θ_L that generates a current token y_i based on a context of the previous $i-1$ tokens, multiple y constitute a function f and user manual (includes rule explanations, code comments, corresponding invocation stages, and so on). In addition, a collection of f and u forms the function base D_f.

3.2 Retriever

For the retriever $p_{\theta_R}(f|x, s)$, we have proposed two solutions tailored for function base of different scales. When the scale of the function base is large, we follow prior work [Lewis *et al.*, 2020] to implement retrieval component based on DPR [Karpukhin *et al.*, 2020], and retriever $p_{\theta_R}(f|x, s)$ follows:

$$p_{\theta_R}(f|x, s) \propto exp(d(f)^{\top}, q(x, s)) \to f_c = [f_1, \cdots, f_K] \tag{2}$$

where s is current states, $d(f)$ is a dense representation of functions (including code comments and rule descriptions), and $q(x, s)$ is a query representation. Retriever $p_{\theta_R}(f|x, s)$ will return a top-K list f_c, where the K functions with highest prior probability. As for small-scale function base, we will fine-tune S-LLMs so that S-LLMs can directly select and utilize the appropriate functions from base D_f to make decisions.

3.3 S-LLMs

We fine-tune S-LLMs to enable them to comprehend the functionality and the appropriate invocation timing of different functions. S-LLMs first select a function f_j from D_f or f_c for stage j:

$$p_{\theta_S}(f_j|x, s) = max([p_{\theta_S}(f_1|x, s), \cdots, p_{\theta_S}(f_K|x, s)]) \tag{3}$$

Then, function f_j will be executed to obtain the intermediate result of the j-th stage:

$$o_j = f_j(x, s), \quad s = o_{j-1} \tag{4}$$

If there are J stages in a step of the task, S-LLMs will select J functions, and the decision-making outputs for that step will be:

$$O = o_J = f_J(x, s), \quad s = o_{J-1} \tag{5}$$

If O meets the requirements of the task, the decision-making process will be halted. Otherwise, O will be regarded as input for the next step.

3.4 Emergency handling of S-LLMs in LD

An advantage of LLMs is their ability to respond to various situations. When using LLMs to control embodied agents, they can respond to unforeseen circumstances. For instance, when LLMs control unmanned vessels for maritime exploration, they might navigate from the open sea to archipelagos, and LLMs can analyze the specific terrain, enabling swift traversal of the archipelago.

Typically, actions required in an emergency situation (such as avoiding whirlpools when controlling unmanned vessels) are simpler compared to the initial various instructions and rules. Therefore, in the LD framework, S-LLMs will transform emergency x_E into functions and add them to the function candidate list f_c:

$$p_{\theta_S}(f_E, u|x_E, s) = \prod_i p_{\theta_S}(y_i|x_E, s, y_{1:i-1}), f_E, u = \prod_i y_i \tag{6}$$

Through Equation 6, S-LLMs will possess stronger general capabilities.

It should be noted that, in contrast to KD, which necessitates S-LLMs to memorize massive L-LLMs' outputs, LD merely requires S-LLMs to remember the usage of functions. Consequently, LD preserves more general capabilities of LLMs, including function generation. The proposed LD is summarized in Algorithm 1.

3.5 Why Selection Is Better

For the aforementioned limitation of S-LLMs, ineffectiveness in following complex instructions, we propose change the function of S-LLMs from generation to selection. Specifically, S-LLMs are required to select the appropriate functions from a provided set, which are to be employed at various stages when confronting a particular problem. This section theoretically analyzes the advantages of selection over generation.

Assuming that the token list of LLMs contains a total of M types of tokens, the retriever provides K types of functions. For generation, the entropy of the prediction is:

$$H_{generation} = -\sum_{i=1}^{M} p'_{\theta_S}(t_i) \log p'_{\theta_S}(t_i), \qquad \sum_{i=1}^{M} p'_{\theta_S}(t_i) = 1 \tag{7}$$

where t_i is a token in the token list.

Algorithm 1 Logic Distillation

```
**Input**: rules (instructions) $x$.
**Parameter**: L-LLMs $p_{\theta_L}$, S-LLMs $p_{\theta_S}$, retriever $p_{\theta_R}$.
**Output**: the decision-making outcome $[o_1, o_2, \cdots]$.
 1: Generate functions $f$ and corresponding user manual $u$ with L-LLMs by Equation 1.
 2: Building function base $D_f$ with $f$ and $u$.
 3: Initialize $O$, $s$.
 4: while Decision-making output $O$ of one step does not meet the task requirements do
 5:     while $j$ in $1, 2, \cdots, J$ do
 6:         Retrieve top-$K$ functions $[f_1, \cdots, f_K]$ with $p_{\theta_R}$, $x$ and $s$ by Equation 2.
 7:         S-LLMs select the most suitable function $f_j$ from $[f_1, \cdots, f_K]$ for stage $j$.
 8:         Obtaining intermediate results $o_j$ by Equation 4.
 9:     end while
10:     $O, s = o_J$
11:     if emergencies then
12:         Generate functions $f_E$ by Equation 6 and add $f_E$ into $[f_1, \cdots, f_K]$.
13:     end if
14: end while
```

With Lagrange multiplier method [Liu, 1972], we get:

$$Q\left(p'_{\theta_S}(t_1), p'_{\theta_S}(t_2), \ldots, p'_{\theta_S}(t_M), \lambda\right) = -\sum_{i=1}^{M} p'_{\theta_S}(t_i) \log p'_{\theta_S}(t_i) + \lambda \left(\sum_{i=1}^{M} p'_{\theta_S}(t_i) - 1\right) \tag{8}$$

then partially differentiating Q in Equation 8 with respect to $p'_{\theta_S}(t_i)$ and λ,

$$\begin{cases} \dfrac{\partial Q}{\partial p'_{\theta_S}(t_i)} = -\log p'_{\theta_S}(t_i) - 1 + \lambda \\ \dfrac{\partial Q}{\partial \lambda} = \sum_{i=1}^{M} p'_{\theta_S}(t_i) - 1 \end{cases} \tag{9}$$

Let Equation 9 be 0, we can get:

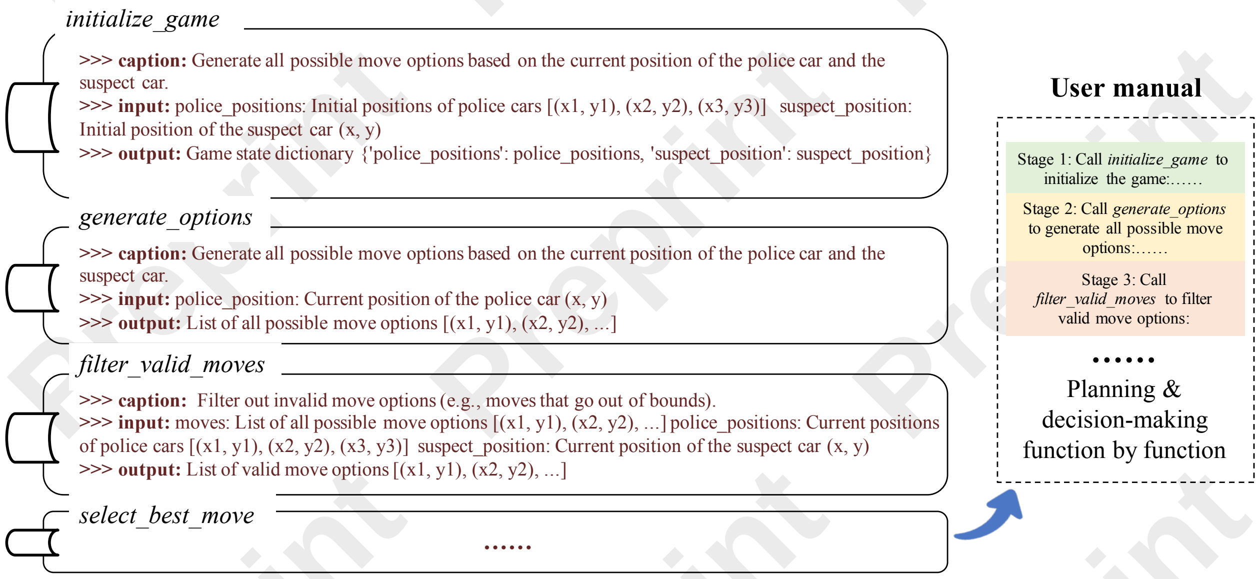
$$p'_{\theta_S}(t_1) = p'_{\theta_S}(t_2) = \ldots = p'_{\theta_S}(t_M) = \frac{1}{M}, \qquad H_{generation} = \log M \tag{10}$$

which is the maximum value of $H_{generation}$.

When we perform selection, the number of candidates will be K, and the maximum value of $H_{selection}$ will be $\log K$. As $K << M$, $\log K << \log M$, the maximum value of $H_{selection}$ will be much smaller than that of $H_{generation}$, and the lower bound of selection will be much higher. Therefore, compared to generation, selection is more effective in maintaining stable outputs for S-LLMs.

4 Experiments

Our experiments aim to: (1) verify the effectiveness of LD in decision-making tasks, (2) explore the reasons for the effectiveness of LD (3) verify that LD has a stronger ability



initialize_game

>>> **caption:** Generate all possible move options based on the current position of the police car and the suspect car.
>>> **input:** police_positions: Initial positions of police cars [(x1, y1), (x2, y2), (x3, y3)] suspect_position: Initial position of the suspect car (x, y)
>>> **output:** Game state dictionary {'police_positions': police_positions, 'suspect_position': suspect_position}

generate_options

>>> **caption:** Generate all possible move options based on the current position of the police car and the suspect car.
>>> **input:** police_position: Current position of the police car (x, y)
>>> **output:** List of all possible move options [(x1, y1), (x2, y2), ...]

filter_valid_moves

>>> **caption:** Filter out invalid move options (e.g., moves that go out of bounds).
>>> **input:** moves: List of all possible move options [(x1, y1), (x2, y2), ...] police_positions: Current positions of police cars [(x1, y1), (x2, y2), (x3, y3)] suspect_position: Current position of the suspect car (x, y)
>>> **output:** List of valid move options [(x1, y1), (x2, y2), ...]

select_best_move

......

User manual

Stage 1: Call *initialize_game* to initialize the game:......

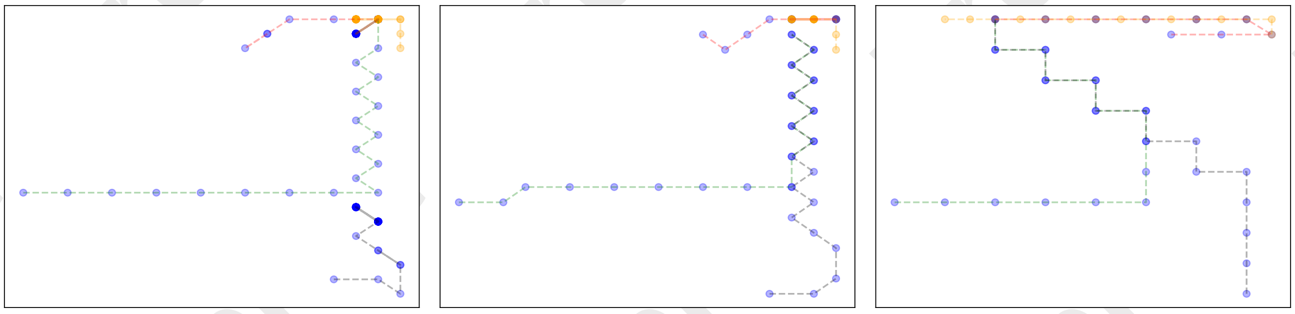
Stage 2: Call *generate_options* to generate all possible move options:......

Stage 3: Call *filter_valid_moves* to filter valid move options:

......

Planning & decision-making function by function

Figure 4: Function base. The L-LLM decomposes the rules and instantiates the decision-making logic into multiple functions (each function performs a specific task, such as calculating distance, etc.). In addition, the L-LLM enables the S-LLM to accurately invoke relevant functions by creating a user manual (including explanations, function comments, corresponding invocation stages, etc.), thereby completing the decision-making process.



(a) S-LLM with KD (Failure) (b) L-LLM (18 steps) (c) S-LLM with LD (15 steps)

Figure 5: The global perspective of the pursuit game based on GLM4 and GLM4-9B. The trajectories of the three blue dots are represented by green, red, and grey dashed lines, respectively, with the darker colors indicating a higher number of passages.

to respond to emergencies. In our experiments, the L-LLMs are GLM-4, LLaMA3-70B and Qwen2.5-72B , while the S-LLMs are GLM4-9B, LLaMA3-7B and Qwen2.5-7B.

4.1 Better Performance in Pursuit Game

We first conduct experiments based on the pursuit game to demonstrate that the proposed method can effectively enhance the decision-making capabilities of S-LLMs. More specifically, the pursuit game involves two sides, each controlled by a different LLM. One LLM manages three blue dots, while the other one controls an orange dot. Each interaction between the two sides constitutes a step. In each iteration, the blue dots are constrained to move by two units, while the orange dot is restricted to a single unit of movement. The game concludes when the Manhattan distance between all three blue dots and the orange dot is less than 2 units.

Specifically, the orange dot is consistently controlled by the original S-LLM, while the blue dots are managed by the L-LLM, S-LLM, S-LLM with KD, and S-LLM with LD, respectively. Additional selection and judgment rules have been introduced to improve the success rate of the baselines. For selection, LLMs are provided with the next decision coordinates to choose from. Regarding judgment, if LLMs make more than seven illegal choices, the game is considered a failure. The upper limit for the number of moves in the game is capped at 100. To perform KD, we initialize 221 sets of starting positions randomly and produce 103,355 sets of outputs with the L-LLM. Subsequently, we fine-tune the S-LLM with LoRA [Hu *et al.*, 2021] based on these outputs. For LD, as depicted in Figure 4, we initially establish a function base with the L-LLM, where L-LLM decomposes the rules and instantiating the decision-making logic into multiple functions. Moreover, to assist the S-LLM in learning how

Methods	Success	Failure without Violation	Failure with Violation	Average Steps of Success
GLM4 (Large)	96.00%	4.00%	0.00%	14.22 steps
GLM4-9B (Small)	0.00%	0.00%	100.00%	—
GLM4-9B-KD	88.50%	10.50%	1.00%	15.23 steps
GLM4-9B-LD	**100.00%**	**0.00%**	**0.00%**	**13.26 steps**
LLaMA3-70B (Large)	94.00%	2.00%	2.00%	16.37 steps
LLaMA3-7B (Small)	0.00%	2.00%	98.00%	—
LLaMA3-7B-KD	80.50%	11.50%	8.00%	18.74 steps
LLaMA3-7B-LD	**100.00%**	**0.00%**	**0.00%**	**15.28 steps**
Qwen2.5-72B (Large)	95.50%	4.00%	0.50%	13.92 steps
Qwen2.5-7B (Small)	1.00%	0.00%	99.00%	56.85 steps
Qwen2.5-7B-KD	89.00%	6.00%	5.00%	14.99 steps
Qwen2.5-7B-LD	**100.00%**	**0.00%**	**0.00%**	**13.23 steps**

Table 1: Results of pursuit game. In this context, "Success" refers to the successful conclusion of the game, where the blue dot captures the orange dot. "Failure without Violation" indicates an unsuccessful outcome due to the inability to capture the orange dot within the specified number of moves. On the other hand, "Failure with Violation" signifies an unsuccessful outcome resulting from a violation of the game rules. Lastly, "Average Steps of Success" quantifies the average number of moves required for successful completion of the game. All methods are tested on 200 sets of starting positions.

to utilize various functions and decide when to invoke them, L-LLM creates a user manual for these functions. By fine-tuning S-LLM with the user manual, it can comprehend the logic of the L-LLM and execute decision-making processes function by function.

The results of the pursuit game are presented in Table 1. It is evident that S-LLM struggles to comprehend complex instructions, as its failures mainly stem from rule violations. Conversely, the "Failure with Violation" rates of L-LLMs are close to zero., demonstrating its superior comprehension and capability to follow instructions. By employing KD to mimic L-LLMs' outputs, S-LLMs' decision-making capabilities significantly improve. Nonetheless, their success rates are still notably lower than that of L-LLMs, and the number of steps in successful instances are higher. As for LD, the game's success rates have reached 100%, and the average number of steps taken by S-LLMs is fewer than that of L-LLMs. These results comprehensively demonstrate the effectiveness of LD in enhancing S-LLMs' decision-making capabilities.

4.2 Why LD is Better

In Figure 5, we initialize different LLMs from identical starting positions: blue dots at $(3, 8), (14, 19), (17, 2)$, and the orange dot at $(20, 18)$ to enable a global comparison of the overall decision-making capabilities among different LLMs.

In Figure 5(a), as S-LLM with KD merely mimics the outputs of L-LLM, blue dots may represent outputs from the L-LLM in different scenarios, resulting in behaviors such as repetitive circling. For instance, the point along the grey trajectory continuously shuttle back and forth, making it impossible to catch up with the orange point. In Figure 5(b), the point controlled by L-LLM appears to backtrack, mainly because of the orange point's continuous back-and-forth movements in an attempt to escape encirclement. Besides, from Figure 5(c), it can be observed that S-LLM with LD enables the blue dots to approach the orange dot in a more direct manner. Thus, contrasting with L-LLM, S-LLM with LD requires fewer steps to successfully capture the target. Such results stem from different emphases: For LD, L-LLM employs a global perspective to design functions, causing the S-LLM to pay more attention to the overall distance from the orange dot. However, during one decision-making step, L-LLM is more susceptible to the influence of the current state, consequently neglecting the global planning. This is also why the performance of LD in Table 1 is superior to that of L-LLM.

4.3 Pursuit Game with Emergencies

In order to assess the capacity of different LLMs to handle emergencies, we introduced a 5×5 restricted area within the game plane. Related results based on GLM4 and GLM4-9B are presented in Table 2.

In the more intricate scenario, L-LLM can still grasp the rules through simple textual descriptions and implement effective decision-making, as the success rate achieve 90%. Conversely, S-LLM with KD achieves a success rate of 52%, with failures resulting from rule violations (accounting for 45.5%), indicating an inability of S-LLM with KD to comprehend the new rules. As for S-LLM with LD, the process involves S-LLM initially abstracting the restricted area as a coordinate filtering function. Then, this function will handle the output of $filter\_valid\_moves$ from Figure 4, producing a list that excludes the coordinates of the restricted area. Subsequently, this list is utilized by $select\_best\_move$ to generate a suitable coordinate. The success rate of S-LLM with LD exceeds that of L-LLM by 10%, and its average number of steps is approximately 4 steps fewer than the baselines, demonstrating the powerful general capabilities of LD. It should be noted that, compared to KD, LD only uses a few functional examples to fine-tune the S-LLM, enabling S-LLM to retain more general capabilities.

In Figure 6, we illustrate the comprehensive decision-making processes of L-LLM and S-LLM with LD, which further validates LD has a stronger ability to respond to emergencies.

This experiment also highlights the advantages of controlling embodied agents through LLMs. Traditional reinforcement learning methods for controlling embodied agents require retraining the model from scratch when encountering

Methods	Success	Failure without Violation	Failure with Violation	Average Steps of Success
GLM4 (Large)	90.00%	7.50%	2.50%	18.98 steps
GLM4-9B (Small)	0.00%	0.00%	100.00%	—
GLM4-9B-KD	52.00%	2.50%	45.50%	18.5 steps
GLM4-9B-LD	**100.00%**	**0.00%**	**0.00%**	**14.65 steps**

Table 2: Results of pursuit game with emergencies.

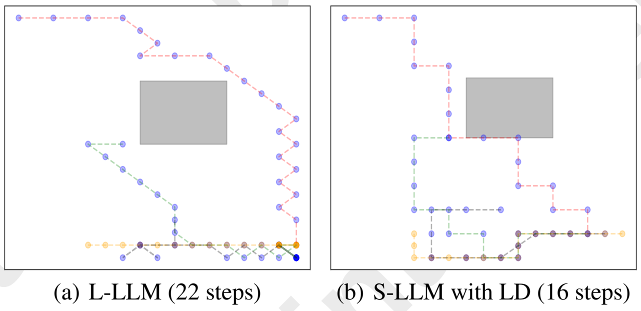


Figure 6: The process of the pursuit game with emergencies.

Methods	Victory	Loss	Draw
GLM4 (Large)	55.50%	42.00%	2.50%
GLM4-9B (Small)	36.50%	62.00%	1.50%
GLM4-9B-KD	48.00%	49.00%	3.00%
GLM4-9B-LD	**59.50%**	**38.50%**	**2.00%**
LLaMA3-70B (Large)	57.00%	42.00%	1.00%
LLaMA3-7B (Small)	39.50%	60.00%	0.50%
LLaMA3-7B-KD	48.50%	58.00%	3.50%
LLaMA3-7B-LD	**58.50%**	**38.50%**	**3.00%**
Qwen2.5-72B (Large)	50.50%	46.50%	3.00%
Qwen2.5-7B (Small)	34.00%	65.50%	0.50%
Qwen2.5-7B-KD	44.50%	54.50%	0.00%
Qwen2.5-7B-LD	**55.00%**	**43.00%**	**2.00%**

Table 3: The victories, losses, and draws of each LLM in 200 times of 21 points.

new influencing factors. In contrast, under the LD framework, S-LLMs can directly accept new rules and add them to the function library, enabling the agent to address new challenges by combining different functions.

4.4 Better Performance in 21 Ponits

To further validate the improvement in S-LLM's decision-making capabilities with LD, we engage LLMs in a modified version of the game "21 points." As depicted in Figure 7, during each round, the two participating LLMs must decide whether to "stand" or "hit", with the objective of surpassing their opponent's total score without exceeding 21 points. Notably, cards "10", "J", "Q", and "K" are each assigned a value of 10, and card "A" can be used as either 1 or 11. Furthermore, we streamline the game by reducing the deck to 26 cards of two suits, thereby enabling the LLMs to deduce the likelihood of exceeding 21 points after requesting a card, based on the available information.

Related results are presented in Table 3. It can be observed

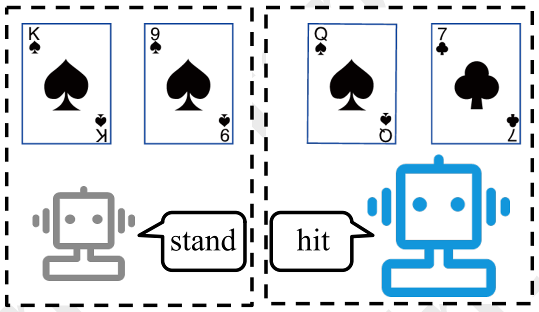


Figure 7: KD vs LD. KD aims to have smaller models mimic the output of larger models, while LD tries to enable smaller models to understand how larger models accomplish a task.

that L-LLMs demonstrate significant advantages in decision-making, as their win rates substantially exceed their rates of losses and draws. As for S-LLMs with KD, we first fine-tune S-LLMs with the outputs of L-LLMs. Although KD enhances S-LLMs' decision-making capabilities, the improvement is not significant, and there is still a gap between S-LLMs and L-LLMs. In contrast, S-LLMs with LD achieve comparable or even superior results compaed to L-LLMs. This is because L-LLMs teach the logic of decision-making to S-LLMs in the form of functions, enabling S-LLMs to maintain a global perspective. However, L-LLMs are susceptible to the influence of the current state in the decision-making process at each step, thereby losing its ability for global planning.

5 Conclusion

LLMs have been widely applied across many different fields. However, larger LLMs (L-LLMs) with powerful capabilities are difficult to deploy on the vast majority of devices due to their parameter scale. In contrast to L-LLMs, smaller open-source LLMs (S-LLMs) are easier to deploy but fall significantly short in performance compared to their larger counterparts. To improve the performance of S-LLMs, researchers have proposed various Knowledge Distillation (KD) methods. Nevertheless, KD merely enables S-LLMs to mimic the outputs of L-LLMs, which is insufficient for addressing decision-making problems. Thus, we propose Logic Distillation (LD), a method that instantiates the logic of L-LLMs by converting instructions into functions, thereby establishing a function base. Subsequently, through fine-tuning, S-LLMs will comprehend the usage of each function, enabling S-LLMs to make decisions. Experimental results demonstrate that the proposed method, with a small amount of example fine-tuning, can enable S-LLMs to match or even surpass the decision-making capabilities of L-LLMs.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant Nos. 62436007, 62325602, 62036010, 62276238, U24A20326), the Foundation of Henan Educational Committee under Grant (No.25HASTIT034), the Natural Science Foundation of Henan under Grant (No.232300421095), the Key R&D Projects in Zhejiang Province (No.2024C01106, 2025C01030), the Zhejiang NSF (LRG25F020001).

References

[Achiam *et al.*, 2023] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[Agarwal *et al.*, 2024] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

[Biswas, 2023] Som S Biswas. Role of chat gpt in public health. *Annals of biomedical engineering*, 51(5):868–869, 2023.

[Bommasani *et al.*, 2021] 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.

[Chen *et al.*, 2021] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[Chen *et al.*, 2024a] Dong Chen, Shuo Zhang, Yueting Zhuang, Siliang Tang, Qidong Liu, Hua Wang, and Mingliang Xu. Improving large models with small models: Lower costs and better performance. *arXiv preprint arXiv:2406.15471*, 2024.

[Chen *et al.*, 2024b] Dong Chen, Yueting Zhuang, Shuo Zhang, Jinfeng Liu, Su Dong, and Siliang Tang. Data shunt: Collaboration of small and large models for lower costs and better performance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11249–11257, 2024.

[Chen *et al.*, 2025] Dong Chen, Zhengqing Hu, Peiguang Fan, Yueting Zhuang, Yafei Li, Qidong Liu, Xiaoheng Jiang, and Mingliang Xu. Kka: Improving vision anomaly detection through anomaly-related knowledge from large language models. *arXiv preprint arXiv:2502.14880*, 2025.

[Chowdhery *et al.*, 2022] 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. *arXiv preprint arXiv:2204.02311*, 2022.

[Dai *et al.*, 2023] Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, et al. Auggpt: Leveraging chatgpt for text data augmentation. *arXiv preprint arXiv:2302.13007*, 2023.

[Feng *et al.*, 2020] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

[GLM *et al.*, 2024] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools, 2024.

[Gou *et al.*, 2021] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[Gu *et al.*, 2024] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[Guo *et al.*, 2024] Junhao Guo, XueFeng Shan, Guoming Wang, Dong Chen, Rongxing Lu, and Siliang Tang. Heart: Heart expert assistant with retrieval-augmented. In *AAAI 2024 Spring Symposium on Clinical Foundation Models*, 2024.

[He *et al.*, 2023] Xingwei He, Zhenghao Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, Weizhu Chen, et al. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*, 2023.

[Hsieh *et al.*, 2023] Cheng-Yu Hsieh, Chun-Liang Li, Chihkuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, 2023.

[Hu *et al.*, 2021] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models.

In *International Conference on Learning Representations*, 2021.

[Karpukhin *et al.*, 2020] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen Tau Yih. Dense passage retrieval for open-domain question answering. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pages 6769–6781. Association for Computational Linguistics (ACL), 2020.

[Kasneci *et al.*, 2023] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

[Koundinya Gundavarapu *et al.*, 2024] Saaketh Koundinya Gundavarapu, Shreya Agarwal, Arushi Arora, and Chandana Thimmalapura Jagadeeshaiah. Machine unlearning in large language models. *arXiv e-prints*, pages arXiv–2405, 2024.

[Lewis *et al.*, 2020] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[Liu *et al.*, 2023] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

[Liu, 1972] I-Shih Liu. Method of lagrange multipliers for exploitation of the entropy principle. *Archive for Rational Mechanics and Analysis*, 46:131–148, 1972.

[Mitra *et al.*, 2023] Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, et al. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*, 2023.

[Mukherjee *et al.*, 2023] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*, 2023.

[Ouyang *et al.*, 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[Peng *et al.*, 2023] Cheng Peng, Xi Yang, Aokun Chen, Kaleb E Smith, Nima PourNejatian, Anthony B Costa, Cheryl Martin, Mona G Flores, Ying Zhang, Tanja Magoc, et al. A study of generative large language model for medical research and healthcare. *NPJ digital medicine*, 6(1):210, 2023.

[Thirunavukarasu *et al.*, 2023] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[Thoppilan *et al.*, 2022] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[Touvron *et al.*, 2023] 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.

[Wu *et al.*, 2023] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

[Xi *et al.*, 2023] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

[Xu *et al.*, 2024] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.

[Zhao *et al.*, 2023] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.