TUNING-FREE ACCOUNTABLE INTERVENTION FOR LLM DEPLOYMENT - A METACOGNITIVE APPROACH

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) have catalyzed transformative advances across a spectrum of natural language processing tasks through few-shot or zero-shot prompting, bypassing the need for parameter tuning. While convenient, this modus operandi aggravates “hallucination” concerns, particularly given the enigmatic “black-box” nature behind their gigantic model sizes. Such concerns are exacerbated in high-stakes applications (e.g., healthcare), where unaccountable decision errors can lead to devastating consequences. In contrast, human decision-making relies on nuanced cognitive processes, such as the ability to sense and adaptively correct misjudgments through conceptual understanding. Drawing inspiration from human cognition, we propose an innovative metacognitive approach, dubbed CLEAR, to equip LLMs with capabilities for self-aware error identification and correction. Our framework facilitates the construction of concept-specific sparse subnetworks that illuminate transparent decision pathways. This provides a novel interface for model intervention after deployment. Our intervention offers compelling advantages: (i) at deployment or inference time, our metacognitive LLMs can self-consciously identify potential mispredictions with minimum human involvement, (ii) the model has the capability to self-correct its errors efficiently, obviating the need for additional tuning, and (iii) the rectification procedure is not only self-explanatory but also user-friendly, enhancing the interpretability and accessibility of the model. By integrating these metacognitive features, our approach pioneers a new path toward engendering greater trustworthiness and accountability in the deployment of LLMs.

1 Introduction

Natural language processing (NLP) has undergone significant advances in recent years, primarily fueled by the advent of Large Language Models (LLMs) (Raffel et al., 2020; Zhou et al., 2022b; OpenAI, 2023). Despite their laudable achievements, LLMs are not infallible; they err due to factors like “hallucination” (McKenna et al., 2023). These vulnerabilities pose critical challenges for the trustworthy deployment of LLMs in high-stakes settings where errors can precipitate significant repercussions. For example, in the application of LLM-assisted medical diagnoses (Monajatipoor et al., 2022), a single misdiagnosis can inflict profound physical and financial costs on the patient.

Despite its significance, the current literature lacks an effective approach to LLM intervention after deployment to help the model overcome those errors. One intuitive method, few-shot or zero-shot prompting (Wei et al., 2022; OpenAI, 2023) recently has shown promising results. Users can directly query LLMs and point out their mistakes using usually “hand-crafted” prompts. Though they are simple, the post-prompting performance remains uncertain. Moreover, it necessitates human expertise both for error identification and prompt design. (2) Another potential method is to fine-tune part of the parameters in LLMs (e.g., the final layers) on erroneously predicted examples (Hardt & Sun, 2023). Besides costly human involvement, this method risks model overfitting on those examples and “catastrophic forgetting” of prior knowledge. (3) Some initial work (Li et al., 2023) repetitively performs activation-level intervention on all examples to get better performance, thus resulting in drastically inflated inference latency.

Against this backdrop, we trifurcate the challenges for LLM intervention into three folds. Firstly, the “black-box” nature of LLMs obscures the malfunction source within the multitude of parameters,
complicating targeted intervention. Secondly, rectification typically depends on domain experts to identify errors, hindering scalability and automation. Thirdly, the architectural complexity and sheer size of LLMs render pinpointed intervention an overwhelmingly daunting task.

In this paper, we advocate that an ideal intervention should be metacognitive, where LLMs are capable of self-aware error identification and correction. This perspective is informed by several key insights from cognitive science literature: (a) Cognitive Perception of Concepts - humans demonstrate the ability to swiftly identify and rectify judgment errors by perceptively recognizing essential features, or “concepts” (Malafouris, 2013; Koh et al., 2020). This ability to hone in on vital features underscores the efficiency of human cognitive processes. (b) Neural Sparsity for Efficiency - building upon the notion of efficiency, the architecture of the human brain provides a valuable lesson. The distribution of neural connections and activity patterns in our brains is characterized by a high degree of sparsity (Gerum et al., 2020). This sparse configuration is believed to facilitate rapid cognitive responses. (c) Conscious Anomaly Detection - human brain exhibits an intrinsic ability to consciously identify anomalies or challenging problems (Penfield, 2015). Upon encountering such situations, it channels additional neural resources to address them effectively.

Building on this premise, we propose an avant-garde Concept-Learning-Enabled metACognitive inteRvention framework, herein termed CLEAR, for LLM deployment. CLEAR facilitates LLMs in mastering concept-specific sparse subnetworks. These subnetworks elucidate transparent decision-making pathways, thereby providing a unique interface for surgical model intervention, that automatically allocates more sparse computing modules to potentially more challenging instances. Distinctively, our approach simultaneously tackles the challenges highlighted above through the following four core contributions:

⭐ Metacognition. At deployment (or inference) time, our metacognitive framework autonomously detects potential mispredictions by measuring logit entropy in pivotal intermediate layers.

⭐ Interpretability. Leveraging the transparency of decision pathways, our CLEAR allows for a logical backtrack to the input, thereby aiding user comprehension and fostering trust in the model.

⭐ Efficiency. Upon identification of a misprediction, the LLM architecture dynamically activates extra internal experts to refine concept perception without necessitating further parameter tuning.

⭐ Effectiveness. Rigorous experiments on real-world datasets with LLM backbones in various sizes and architectures manifest that our intervention consistently improves inference-time predictions.

2 RELATED WORK

Intervention on Deep Models for Error Mitigation. Historically, error mitigation in machine learning emphasized simpler models, such as Decision Trees and Random Forests, where corrections were largely heuristic and human-driven (Doshi-Velez & Kim, 2017). With the evolution of machine learning techniques, there was a pivot towards leveraging algorithms themselves for error detection, emphasizing the removal of non-relevant data and unveiling crucial fault-application relationships (Abich et al., 2021). The ascendance of neural networks, and LLMs in particular, brought forth new intervention paradigms. Fine-tuning emerged as a primary strategy for addressing model shortcomings, despite its challenges related to overfitting and catastrophic forgetting of prior knowledge (Wang et al., 2019; French, 1999). Few-shot and Zero-shot prompting marked another avenue, guiding models without altering their internal makeup, leading to inherent limitations in error repeatability (Wei et al., 2022; OpenAI, 2023). Deeper interventions, targeting model architectures, have delivered promising accuracy, yet with computational trade-offs (Li et al., 2023). Notably, quantum error mitigation approaches, though out of our current scope, underline the breadth of exploration in this domain (Subramanian Ravi et al., 2021).

Concurrently, the push towards model interpretability has intensified (Carvalho et al., 2019; Koh et al., 2020; Yukselkongul et al., 2022). The ultimate goal is to design systems whose inner workings can be easily understood, thereby facilitating targeted interventions. Such transparency is indispensable in critical sectors like healthcare, demanding specialized interventions that are usually hand-carfted by domain experts (Farrell, 2021; Monajatipoor et al., 2022).
Under review as a conference paper at ICLR 2024

Figure 2: The illustration of the proposed framework CLEAR, comprised of two components: (a) Concept Learning, where the LLM backbone learns to construct concept-specific sparse networks via MoCE; and (b) Metacognitive Intervention, which involves logit entropy scrutiny, dynamic expert allocation, and pseudo intervention, and offers retrospective accountability.

Metacognitive Approaches. Metacognition, eloquently described as “thinking about thinking”, has long been acknowledged in cognitive science (Flavell, 1979), resonating through educational and clinical paradigms (Zimmerman, 2013; Moritz & Woodward, 2007). This foundational knowledge has segued into AI, aspiring towards machines with self-reflective and adaptive capabilities (Cox, 2005). Recent endeavors have strived to infuse cognitive inspirations into models, emphasizing a deeper “understanding” of their decisions (Malafouris, 2013). However, genuinely metacognitive LLMs remain an elusive goal, with inherent challenges arising from their black-box nature and their vast, intricate architectures.

3 METHODOLOGY

The proposed Concept-Learning-Enabled Metacognitive Intervention framework, CLEAR is comprised of two crucial components: (1) Concept Learning: the learning of concept-specific sparse subnetworks for LLMs. (2) Metacognitive Intervention: automatic error identification and rectification. We provide their detailed elaborations below.

3.1 CONCEPT LEARNING FOR LARGE LANGUAGE MODELS

Basic Setup. Our primary focus is the enhancement of Large Language Models (LLMs) within the realm of text classification tasks during the inference phase. Given a dataset $D = \{(x^{(i)}, y^{(i)}, c^{(i)})\}_{i=1}^{N}$, we utilize an LLM, denoted by $f_{\theta}$, to transform an input text $x \in \mathbb{R}^{D}$ into a latent space representation $z \in \mathbb{R}^{E}$. This latent representation is then classified via a linear classifier $g_{\phi}$ into the respective task label $y$. Here $\{c^{(i)}\}_{i=1}^{N}$ denotes the critical features, or “concepts” annotated by humans (Koh et al., 2020; Abraham et al., 2022). These concepts are typically represented using one-hot vectors. Take, for instance, a restaurant review sentiment dataset. The concept “Food” might be denoted by the vector $[0, 0, 1]$, signifying a “Positive” sentiment towards the food. The other vector positions represent alternative sentiments, specifically “Negative” and “Unknown”.

Incorporating Concept Bottlenecks for LLMs. Our general pipeline is inspired by a previous work (Koh et al., 2020) on image classifications. Instead of altering LLM encoders—which might
compromise the integrity of the text representation—we incorporate a linear layer, characterized by a sigmoid activation function $p_\psi$. This layer maps the latent representation $z \in \mathbb{R}^K$ to a concept space $c \in \mathbb{R}^L$, thus creating a decision-making pathway depicted as $x \rightarrow z \rightarrow c \rightarrow y$. By allowing for multi-class concepts, we aim to achieve nuanced interpretations. For ease of reference, LLMs integrated with Concept Bottlenecks are termed LLM-CBMs (e.g., BERT-CBM). The training of LLM-CBMs is dual-faceted: (1) Ensure the concept prediction $\hat{c} = p_\psi(f_\theta(x))$ aligns with the input’s true concept labels $c$. (2) Ensure the label prediction $\hat{y} = g_\phi(p_\psi(f_\theta(x)))$ corresponds with true task labels $y$. Our framework predominantly employs the joint training approach due to its superior performance, as supported by Anonymous (2023). The joint training mechanism harmonizes the concept encoder and label predictor through a weighted sum, represented as $L_{\text{joint}}$:

$$
\theta^*, \psi^*, \phi^* = \arg\min_{\theta, \psi, \phi} L_{\text{joint}}(x, c, y) = \arg\min_{\theta, \psi, \phi} \left[ L_{\text{CE}}(g_\phi(p_\psi(f_\theta(x)), y) + \gamma L_{\text{CE}}(p_\psi(f_\theta(x)), c)) \right] = \arg\min_{\theta, \psi, \phi} \sum_{k=1}^K \left[ L_{\text{CE}}(g_{\phi_k}(p_{\psi_k}(f_\theta(x)), y) + \gamma L_{\text{CE}}(p_{\psi_k}(f_\theta(x)), c_k)) \right],
$$

(1)

where, $L_{\text{CE}}$ represents the Cross-Entropy loss. The third line of the equation incorporates the loss iterating across the concepts, a detail that will prove pivotal soon. Notably, the sensitivity of jointly trained LLM-CBMs to the loss weight $\gamma$ requires attention. By default, we set $\gamma$ to 5.0, based on its optimized performance as observed in Anonymous (2023). Further details on varying training strategies are expounded in Appendix A. It should be noted that conventional LLM-CBMs (Koh et al., 2020) tend to train all concepts simultaneously. This concurrent training potentially muddles the parameters meant for individual concept prediction, thus hampering precise intervention.

**Building Concept-Specific Sparse Subnetworks via Mixture of Concept Experts.** Our research presents the Mixture of Concept Experts (MoCE) framework, a novel approach to creating pathways anchored in specific concepts, thereby enhancing targeted interventions. This model takes cues from mixture-of-expert (MoE) paradigms (Shazeer et al., 2017), known for their dynamic activation of unique network subsets per input. By conditioning on concept-based computation, MoCE crafts sparse modules, fine-tuning the encoding of text inputs as per their inherent concepts.

Conforming to conventions, we structure blocks of MoCEs as the expert layer. This layer comprises a multi-head attention block combined with multiple parallel experts. Specifically, we adapt MoCE for Transformer architectures, integrating MoE layers within successive Transformer blocks. Crafting a MoCE expert typically involves segmenting the conventional MLP of transformers into more compact segments (Zhang et al., 2021) or duplicating the MLP (Fedus et al., 2022). It’s noteworthy that the majority of extant MoE studies have predominantly focused on the MLP segment within transformers. This focus arises because MLPs account for approximately two-thirds of the entire model parameter set, serving as key repositories of accrued knowledge within memory networks (Geva et al., 2020; Dai et al., 2022).

The experts can be symbolized as $\{e_m\}_{m=1}^M$, where $m$ signifies the expert index and $M$ is the total count of experts. For each concept $c_k$, an auxiliary routing mechanism, dubbed $r_k(\cdot)$, is deployed. This mechanism identifies the top-$T$ experts based on peak scores $r_k(x)_m$, with $x$ representing the present intermediate input embedding. Generally, $T$ is much smaller than $N$, which underscores the sparse activations among modules of the LLM backbone, making the inference of the model more efficient. The output, $x'$, emanating from the expert layer is:

$$
x' = \sum_{k=1}^K \sum_{m=1}^T r_k(x)_m \cdot e_m(x); \quad r_k(x) = \text{top}-T(\text{softmax}(\zeta(x)), T),
$$

(2)

where $\zeta$ is a shallow MLP representing learnable routers (Fedus et al., 2022). For the $k$th concept, the expert $e_k(\cdot)$ initially processes the given features, after which the router amplifies it using coefficient $r_k(x)_m$. The combined embeddings across concepts yield the output $x'$. The $\text{top}-T$ operation retains the top $T$ values, nullifying the others. Typically, a balancing mechanism, such as load or importance balancing loss (Shazeer et al., 2017), is implemented to avert the risk of representation collapse, preventing the system from repetitively selecting the same experts across diverse inputs. Transitioning to matrix representation for all MoE layers in the LLM structure, we derive:

$$
y = \sum_{k=1}^K \phi_k \cdot \sigma(\psi_k \cdot f_\theta_k(x)) = \sum_{k=1}^K \phi_k \cdot \psi_k \cdot \sum_{m=1}^T R_k(x)_m \cdot E_m(x),
$$

(3)
where $\sigma(\cdot)$ is the sigmoid projector’s activation function, with $R(\cdot)$ and $E(\cdot)$ symbolizing matrix incarnations of all expert layer routers and experts. Crucially, equation 3 portrays a factorized decision trajectory, streamlining the classification framework. This can be optimized through a single backward iteration of the composite loss as outlined in equation 2. Note that equation 3 accomplishes a core objective: during inference, the LLM backbone’s final classifications intrinsically rely on the learned routing policies, the chosen experts, and the perceived concepts. This unique accountability offers an interface for precise error identification and interventions.

3.2 Tuning-free Metacognitive Intervention

At its core, our metacognitive intervention emulates human cognitive processes: similar to the way human brains discern potential pitfalls or intricate challenges, our CLEAR framework proactively identifies these issues. It then adeptly marshals extra sparse neural resources, specifically experts, to address these challenges. In this subsection, we elucidate how this is realized through our delineated sparse decision pathways, in the form of answering three distinctive research questions.

\begin{center}
\textbf{RQ1: How to achieve “metacognition” for intervention on LLMs?}
\end{center}

\begin{center}
\textbf{A1: By autonomously monitoring anomalous pattern at critical intermediate layers.}
\end{center}

▷ Logit Entropy Scrutiny. The foremost goal is to automatically identify potential errors or more complex cases. As inferred from Equation equation 3, two critical decision-making phases notably impact the ultimate label prediction: (a) the deduced routing $\{R_k(x)\}_{K=1}^{K}$ of the final MoCE layer, and (b) the determined concept activation $\hat{a} = \{\hat{a}_k\}_{K=1}^{K} = \psi \cdot f_\theta(x)$. Intuitively, an elevated entropy of predictive logits denotes a more dispersed distribution over experts or concept options, signifying lower model confidence and pinpointing instances that deserve additional attention. For this purpose, the Shannon entropy is utilized for logits within the routine and concept activation:

$$H(p) = - \sum_{j=1}^{\text{softmax}(l_j)} \log(\text{softmax}(l_j)). \quad (4)$$

For illustration, the distributions of logits and entropy for concept prediction are depicted using kernel density estimation in Figure 3. It is evident that predictions with errors tend to demonstrate lower confidence and augmented entropy, reinforcing our premise. For automation, as we iterate through the concepts, K-Means clustering is employed to divide confidence levels into two clusters (K=2). The subset with lower confidence is considered to stem from the more challenging instances. K-Means offers the advantage of determining thresholds dynamically, eliminating human involvement. If, for a single concept prediction relating to an instance, the confidence levels of both the routine and concept activation surpass the corresponding thresholds, we tag this concept prediction as potentially erroneous. We demonstrate further study on the scrutiny in Figure 6 (a) and (b).

\begin{center}
\textbf{RQ2: Once a potential error is identified during inference, how to intervene on LLMs “without extra parameter tuning”?}
\end{center}

\begin{center}
\textbf{A2: By dynamically allocating experts and enforcing preparatory rehearsal during training.}
\end{center}

▷ Tuning-free Intervention. Once an erroneous prediction is identified, we allocate augmented computational resources to secure a more reliable prediction. This operation can be easily achieved by setting the maximum expert number from $T$ to a larger number $T'$ for the router as below. Note that this operation is very efficient since no extra parameter tuning is involved.

$$r_k(x) = \text{top-T}(\text{softmax}(\zeta(x)), T') \quad (5)$$

▷ Pseudo Intervention during Concept Learning. Both existing research (Chen et al., 2023) and our experiments (Figure 6 (c) and (d)) indicate that directly adding more experts at the inference stage results in marginal improvements. Drawing inspiration from how humans reinforce understanding of challenging subjects through repeated practice before the final examination, we emulate a similar rehearsal mechanism during concept learning for better metacognitive intervention. As the LLM
model is fine-tuned on the task dataset, we progressively raise the count of experts from $T$ to $T'$ linearly after a predetermined number of training epochs, typically post the halfway mark. This strategy of pseudo intervention during the training phase significantly enhances predictions when the expert count is increased during the inference-time metacognitive intervention, as depicted in Figure 6 (c) and (d). Through this essential rehearsal setup, and by sequentially executing the steps outlined in equation 4 and equation 5, the LLM backbone is empowered to autonomously detect possible errors, addressing them more robustly with minimal human oversight.

**RQ3:** How can users understand the intervention?

**A3:** By backtracking from the task label, through the sparse pathway, to the input text.

▷ Retrospective Accountability. A standout feature of our metacognitive intervention is its inherent explicability. Using the decision-making pathways showcased in equation 3, one can trace back from the task label prediction, passing through perceived concepts and activated subnetworks (experts), all the way to the initial text input, as shown in Figure 2. Illustrative examples are provided in Figure 4. The incorporation of our framework, CLEAR, represents a harmonious blend of precision, flexibility, and accountability.

### 4 Experiments

#### 4.1 Experimental Setup

**Datasets.** Our experiments are conducted on two widely-used real-world datasets: CEBaB (Abram et al., 2022) and IMDB-C (Anonymous, 2023). Each of them is a text classification dataset comprised of human-annotated concepts and task labels. Their statistics are presented in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CEBaB (5-way classification)</th>
<th>IMDB-C (2-way classification)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train / Dev / Test</td>
<td>1755 / 1673 / 1685</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Concept</th>
<th>Label</th>
<th>Negative</th>
<th>Positive</th>
<th>Unknown</th>
<th>Label</th>
<th>Negative</th>
<th>Positive</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Food</td>
<td>1693 (33.1%)</td>
<td>2087 (40.8%)</td>
<td>1333 (26.1%)</td>
<td></td>
<td>Acting</td>
<td>76 (38%)</td>
<td>66 (33%)</td>
<td>58 (29%)</td>
</tr>
<tr>
<td>Ambiance</td>
<td>787 (15.4%)</td>
<td>994 (19.4%)</td>
<td>3332 (65.2%)</td>
<td></td>
<td>Storyline</td>
<td>80 (40%)</td>
<td>77 (38.5%)</td>
<td>43 (21.5%)</td>
</tr>
<tr>
<td>Service</td>
<td>1249 (24.4%)</td>
<td>1397 (27.3%)</td>
<td>2467 (48.2%)</td>
<td></td>
<td>Emotional Arousal</td>
<td>74 (37%)</td>
<td>73 (36.5%)</td>
<td>53 (26.5%)</td>
</tr>
<tr>
<td>Noise</td>
<td>645 (12.6%)</td>
<td>442 (8.6%)</td>
<td>4026 (78.7%)</td>
<td></td>
<td>Cinematography</td>
<td>118 (59%)</td>
<td>43 (21.5%)</td>
<td>39 (19.4%)</td>
</tr>
</tbody>
</table>

**Baselines.** In this study, our evaluation primarily involves two categories of frameworks as baselines. For an in-depth analysis, we examine both (a) the performance on the test sets and (b) the performance on the development sets, before and after the intervention. This dual-faceted examination allows us to assess the intervention’s effectiveness and evaluate the model’s potential deterioration in generalizability and catastrophic forgetting of critical prior knowledge. Different LLM backbones are employed in our analysis, including BERT (Devlin et al., 2018), OPT (Zhang et al., 2022), and T5 (Raffel et al., 2020). We adjust our choice of LLM backbone per the specific methods employed:

▷ Direct Intervention Methods: (i) Directly prompting the LLM with human identifying mispredictions. For this method, we use GPT-4 (OpenAI, 2023) as the backbone, as it is widely regarded as the most capable LLM currently. (ii) Directly fine-tuning the LLM backbones on mispredicted instances identified by humans. (iii) Employing the activation-level intervention method, ITI (Li et al., 2023), mentioned in the introduction.

▷ Concept Bottleneck Models (CBMs) support concept-level interventions, but still require human experts to identify mispredictions. We consider the following recent CBM frameworks as baselines: (iv) Vanilla CBMs (Koh et al., 2020) map the text into concepts using the LLM backbone and involve another linear classifier to perform the final classification. (v) Label-free CBMs (LF-CBMs) (Oikarinen et al., 2022) use GPT-4 to obtain the concept labels. (vi) Concept embedding models (CEMs) (Zarlenga et al., 2022) that learn continuous embeddings for concepts.

#### 4.2 Superior Performance of CLEAR

The comparative results are presented in Table 2. Reported scores are the averages of three independent runs. Our work is based on general text classification implementations. The implementation of
Table 2: We compare model performance on the CEBaB and IMDB-C datasets, using Macro F1 as the evaluation metric, expressed in percentages (%). Scores shaded in gray highlight instances where the model experienced catastrophic forgetting, leading to a decline in performance on the development set. Scores shaded in red indicate a decrease in performance following the intervention. Scores shaded in blue are from CLEAR.

| Methods Backbones | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task | Concept Task |
|-------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Prompting GPT4    | 64.32        | 63.82        | 70.32        | 69.52        | 64.32        | 63.82        | 70.32        | 69.52        | 64.32        | 63.82        | 70.32        | 69.52        | 64.32        | 63.82        | 70.32        | 69.52        | 64.32        | 63.82        | 70.32        | 69.52        | 64.32        | 63.82        | 70.32        | 69.52        |
| Fine-tuning BERT  | 84.54        | 77.62        | 76.84        | 75.24        | 84.54        | 77.62        | 76.84        | 75.24        | 84.54        | 77.62        | 76.84        | 75.24        | 84.54        | 77.62        | 76.84        | 75.24        | 84.54        | 77.62        | 76.84        | 75.24        | 84.54        | 77.62        | 76.84        | 75.24        |
| Fine-tuning OPT   | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        |
| Fine-tuning T5    | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        | 85.86        | 78.32        | 76.25        | 74.86        |

Effectiveness. The presented framework, CLEAR, unfailingly surpasses all baseline models in concept prediction and task label prediction, both before and after the intervention. This consistent outperformance underscores the robustness and efficiency of the CLEAR framework across various conditions and parameters. (a) In the concept learning phase, the proposed MoCE layers play a pivotal role. By constructing sparse, concept-specific subnetworks, the MoCE layers facilitate the efficient disentanglement of concepts. This organized division significantly smoothens and enhances the internalization of concepts, laying a solid foundation for further enhancement during the intervention phase. (b) During the intervention phase, the excellence of CLEAR further shines. It elevates prediction accuracy through precisely targeted interventions, tailoring its approach to the specific challenges and complexities encountered in each instance. This meticulous and adaptable strategy allows CLEAR to hone in on and address the unique difficulties faced by each prediction task, ensuring optimal enhancement of prediction accuracy.

Metacognition. Beyond raw performance metrics, the CLEAR framework profoundly underscores its metacognitive prowess, presenting a triumvirate of decisive advantages: efficiency, accountability, and autonomy, setting it distinctly apart from existing baselines. (a) Efficiency: Unlike direct intervention methods, CLEAR is free from extensive tuning, safeguarding it from prevalent issues like catastrophic forgetting encountered in fine-tuning methods (shaded in gray). (b) Autonomy: Distinct from CBMs, CLEAR operates without human intervention, ensuring complete autonomy. This self-sufficiency expands its applicability, particularly in areas where human expertise is limited or costly. Notably, LF-CBMs, utilizing GPT-4 to extract noisy concept labels, display a detrimental effect from intervention (highlighted in pink). This observation further underscores the criticality of accurate and targeted intervention. (c) Accountability: CLEAR provides a comprehensive, multilayered insight into its decision-making process, covering concept, subnetwork, and input levels. This transparency significantly amplifies user trust, offering clarity and assurance in the framework’s operations and decisions. We will go through more details of those advantages in subsequent subsections.

4.3 Extra Investigation and Ablation Study

Accountability. CLEAR does not just execute tasks; it stands out by ensuring retrospective interpretability and in-depth insight into its metacognitive intervention processes. This transparency permeates various levels through backtracking, offering concept-level, subnetwork-level, and input-level explanations. This multilayered insight not only fulfills intellectual curiosity but also enhances user trust and confidence in CLEAR. By understanding the “how” and “why” behind each decision, users gain a more profound insight into the model’s operations, leading to informed and confident interaction with the framework.

The story is not that good... The acting is pretty poor too... manages to work with some very creepy visuals and atmosphere... It’s a quiet kind of horror that isn’t made anymore...

Table 3: Efficiency comparison between interventions

<table>
<thead>
<tr>
<th>Method</th>
<th>Human labels</th>
<th>Parameter tuning</th>
<th>Targeted intervention</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompting</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Fine-tuning</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CBM</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>CLEAR</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 4: Ablation study on intervention mechanism. Scores are reported in %.

<table>
<thead>
<tr>
<th>Methods</th>
<th>CEBaB</th>
<th>JMBaC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre-intervention</td>
<td>Post-intervention</td>
</tr>
<tr>
<td>Concept</td>
<td>Task</td>
<td>Concept</td>
</tr>
<tr>
<td>CLEAR (oracle)</td>
<td>89.63</td>
<td>81.30</td>
</tr>
<tr>
<td>CLEAR</td>
<td>89.63</td>
<td>81.30</td>
</tr>
</tbody>
</table>

Ablation Study. In this section, we perform comprehensive ablation studies to evaluate the critical components of CLEAR, including the intervention mechanism options, logit entropy scrutiny, and pseudo intervention. We will discuss each result in detail.

Case Study. To further illustrate, we present a detailed case study of the metacognitive intervention process in Figure 4. More examples are included in Appendix C. This depiction illuminates the transition of the predicted label for the concept “Cinematography” from incorrect “–” to correct “+”, subsequently refining the final task label. Texts highlighted in red indicates the clues overlooked by insufficient experts. Moreover, by analyzing expert and concept activations before and after the intervention, we reveal the neural mechanics underpinning the intervention strategy at the subnetwork level, offering additional real-world implications. For instance, we can compute the influence $I$ of each concept $c_k$ to the final decision by the product of the concept activation $\hat{a}_k$ and the corresponding weight $w_k$ in the linear classifier: $I(c_k) = \hat{a}_k \cdot w_k$. The results are visualized in Figure 5. This capability to correct and interpret the underlying causes for prediction errors further boosts the model’s overall trustworthiness and usability. Coupled with experimental findings, this case study enriches our understanding of the potential of the proposed metacognitive interventions, showcasing them not just as a model fine-tuning method, but as a structured approach towards more transparent and adaptable AI systems.

Autonomy and Efficiency. CLEAR also demonstrates unique advantages with its full autonomy and tuning-free interventions. We list the comparison of important features among all intervention methods in Table 3. From the comparison, we can observe that CLEAR is the only framework that achieves this impressive enhancement without the need for extensive human involvement or intricate parameter tuning, which are often required by other existing methods. This self-sufficient functionality not only streamlines the operation of the CLEAR framework but also reinforces its reliability and effectiveness. The absence of heavy reliance on human input or complex tuning procedures eliminates potential sources of error and inconsistency, further bolstering the robustness, consistency, and dependability of the CLEAR framework.

Figure 4: Illustration of an case study for the accountable metacognitive intervention.

Figure 5: Contributions of Concepts.
In conclusion, CLEAR stands out as a pioneering framework, uniquely positioned to alleviate the contemporary challenges faced by Large Language Models (LLMs). This paper outlines its robust capabilities in autonomously identifying and correcting errors, thereby reducing the need for extensive human oversight and intricate adjustments. By employing a metacognitive strategy inspired by human cognitive processes, CLEAR enables the construction of transparent, concept-specific sparse subnetworks. This attribute ensures clear, comprehensible decision pathways and eases post-deployment model intervention. In tackling the enduring “black-box” issue prevalent in LLMs, CLEAR confidently showcases its effectiveness in diminishing mispredictions and bolstering overall model interpretability and accessibility. These advances by CLEAR underscore a significant enhancement in both the performance and reliability of LLMs, ensuring their more trustworthy and accountable deployment in diverse real-world scenarios. Moving forward, the widespread adoption of CLEAR promises a tangible, positive shift in the landscape of LLM deployment, underscoring its role as an invaluable asset in the evolution of machine learning models.
REFERENCES


A Definitions of Different Training Strategies

Given a text input $x \in \mathbb{R}^D$, concepts $c \in \mathbb{R}^K$ and its label $y$, the strategies for fine-tuning the text encoder $f_\theta$, the projector $p_\psi$, and the label predictor $g_\phi$ are defined as follows:

i) **Vanilla fine-tuning an LLM:** The concept labels are ignored, and then the text encoder $f_\theta$ and the label predictor $g_\phi$ are fine-tuned either as follows:

$$\theta, \phi = \arg\min_{\theta, \phi} \mathcal{L}_{CE}(g_\phi(f_\theta(x), y),$$

or as follows (frozen text encoder $f_\theta$):

$$\phi = \arg\min_{\phi} \mathcal{L}_{CE}(g_\phi(f_\theta(x), y),$$

where $\mathcal{L}_{CE}$ indicates the cross-entropy loss. In this work we only consider the former option for its significant better performance.

ii) **Independently training LLM with the concept and task labels:** The text encoder $f_\theta$, the projector $p_\psi$, and the label predictor $g_\phi$ are trained separately with ground truth concepts labels and task labels as follows:

$$\theta, \psi = \arg\min_{\theta, \psi} \mathcal{L}_{CE}(p_\psi(f_\theta(x)), c),$$

$$\phi = \arg\min_{\phi} \mathcal{L}_{CE}(g_\phi(c), y).$$

During inference, the label predictor will use the output from the projector rather than the ground-truth concepts.

iii) **Sequentially training LLM with the concept and task labels:** We first learn the concept encoder as the independent training strategy above, and then use its output to train the label predictor:

$$\phi = \arg\min_{\phi} \mathcal{L}_{CE}(g_\phi(p_\psi(f_\theta(x)), y).$$

iv) **Jointly training LLM with the concept and task labels:** Learn the concept encoder and label predictor via a weighted sum $\mathcal{L}_{joint}$ of the two objectives described above:

$$\theta, \psi, \phi = \arg\min_{\theta, \psi, \phi} \mathcal{L}_{joint}(x, c, y)$$

$$= \arg\min_{\theta, \psi, \phi} [\mathcal{L}_{CE}(p_\psi(f_\theta(x)), y)$$

$$+ \gamma \mathcal{L}_{CE}(p_\psi(f_\theta(x)), c)] .$$

It’s worth noting that the LLM-CBMs trained jointly are sensitive to the loss weight $\gamma$. We tune the value for $\gamma$ for better performance (Anonymous, 2023).

B Implementation Detail

In this section, we provide more details on the implementation settings of our experiments. Specifically, we implement our framework with PyTorch (Paszke et al., 2017) and HuggingFace (Wolf et al., 2020) and train our framework on a single 80 GB Nvidia A100 GPU. We follow a prior work (Abraham et al., 2022) for backbone implementation. All backbone models have a maximum token number of 512 and a batch size of 8. We use the Adam optimizer to update the backbone, projector, and label predictor according to Section 3.1. The values of other hyperparameters (Table 5 in the next page) for each specific PLM type are determined through grid search. We run all the experiments on 4 Nvidia A100 GPUs with 80GB RAM.

For the LLM backbones, we use their public versions available on HuggingFace. Specifically, we deploy bert-base-uncased, facebook/opt-350m, and t5-base. In our implementation, we also include other baseline backbones from more language model families. We intentionally include the above three in the main experiment results for their similar sizes. The other backbones include: roberta-base, distilbert-base-uncased, gpt2, facebook/opt-125m, facebook/opt-1.3b, and switch-transformer-base.
Table 5: Key parameters in this paper with their annotations and evaluated values. Note that bold values indicate the optimal ones.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Specification</th>
<th>Definitions or Descriptions</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_len</td>
<td>-</td>
<td>maximum token number of input</td>
<td>128 / 256 / <strong>512</strong></td>
</tr>
<tr>
<td>batch_size</td>
<td>-</td>
<td>batch size</td>
<td>8</td>
</tr>
<tr>
<td>epoch</td>
<td>-</td>
<td>maximum training epochs</td>
<td>30</td>
</tr>
<tr>
<td>lr</td>
<td>DistilBERT</td>
<td>learning rate when the backbone is DistilBERT</td>
<td>1e-3 / <strong>1e-4</strong> / 1e-5 / 1e-6</td>
</tr>
<tr>
<td></td>
<td>BERT</td>
<td>learning rate when the backbone is BERT</td>
<td>1e-3 / <strong>1e-4</strong> / 1e-5 / 1e-6</td>
</tr>
<tr>
<td></td>
<td>RoBERT</td>
<td>learning rate when the backbone is RoBERT</td>
<td>1e-3 / <strong>1e-4</strong> / 1e-5 / 1e-6</td>
</tr>
<tr>
<td></td>
<td>OPT-125M</td>
<td>learning rate when the backbone is OPT-125M</td>
<td>1e-3 / <strong>1e-4</strong> / 1e-5 / 1e-6</td>
</tr>
<tr>
<td></td>
<td>OPT-350M</td>
<td>learning rate when the backbone is OPT-350</td>
<td>1e-4 / <strong>1e-5</strong> / 1e-6 / 1e-7</td>
</tr>
<tr>
<td></td>
<td>OPT-1.3B</td>
<td>learning rate when the backbone is OPT-1.3B</td>
<td>1e-4 / <strong>1e-5</strong> / 1e-6 / 1e-7</td>
</tr>
<tr>
<td></td>
<td>CLEAR</td>
<td>learning rate for CLEAR</td>
<td>1e-4 / <strong>3e-4</strong> / 5e-4 / 7e-4 / <strong>1e-5</strong></td>
</tr>
<tr>
<td>γ</td>
<td>DistilBERT</td>
<td>value of γ when the backbone is DistilBERT</td>
<td>1 / <strong>3</strong> / 5 / <strong>7</strong> / 9</td>
</tr>
<tr>
<td></td>
<td>BERT</td>
<td>value of γ when the backbone is BERT</td>
<td>1 / <strong>3</strong> / 5 / <strong>7</strong> / 9</td>
</tr>
<tr>
<td></td>
<td>RoBERT</td>
<td>value of γ when the backbone is RoBERT</td>
<td>1 / <strong>3</strong> / 5 / <strong>7</strong> / 9</td>
</tr>
<tr>
<td></td>
<td>OPT-125M</td>
<td>value of γ when the backbone is OPT-125M</td>
<td>1 / <strong>3</strong> / 5 / <strong>7</strong> / 9</td>
</tr>
<tr>
<td></td>
<td>OPT-350M</td>
<td>value of γ when the backbone is OPT-350</td>
<td>1 / <strong>3</strong> / 5 / <strong>7</strong> / 9</td>
</tr>
<tr>
<td></td>
<td>OPT-1.3B</td>
<td>value of γ when the backbone is OPT-1.3B</td>
<td>1 / <strong>3</strong> / 5 / <strong>7</strong> / 9</td>
</tr>
<tr>
<td></td>
<td>CLEAR</td>
<td>value of γ for CLEAR</td>
<td>5 / <strong>7</strong> / 9 / <strong>10</strong> / 11 / 13 / 15</td>
</tr>
</tbody>
</table>

C More Examples from Real-World Datasets

Figure 7: An example for the metacognitive intervention on one instance from the CEBaB dataset.

Figure 8: An example for the metacognitive intervention on one instance from the IMDB-C dataset.

D Comparison with Existing Works on MoE for LLMs

Mixture of Experts in Large Language Models. The incorporation of Mixture of Experts (MoE) into Large Language Models (LLMs) has evolved significantly, with early research by Shazeer et al. (2017) laying the groundwork. These foundational studies (Fedus et al., 2022; Zhou et al., 2022a; Du et al., 2022; Artetxe et al., 2021; Shen et al., 2023) focused primarily on improving model performance and computational efficiency in a black-box manner. On the contrary, in this work, we utilize the design of MoE in LLMs for metacognitive capabilities. This novel approach, distinct from earlier efficiency-focused applications, uses MoE for error detection and correction, a critical step towards solving the interpretability and trust issues in AI decision-making. Our framework, CLEAR, contributes to this evolving landscape by embedding MoE within a metacognitive framework, emphasizing error rectification, transparency, and autonomy in LLMs. This shift marks a significant advancement from traditional MoE applications, positioning CLEAR at the forefront of innovative LLM enhancement strategies.
E  ANALYSIS OF K-MEANS IN LOGITS SCRUTINY

Figure 9: Studies on the design of K-means for logits scrutiny. This figure illustrates the effectiveness of K-means in distinguishing between correct and erroneous logits for both routing and concept prediction. Logits are normalized using a softmax function, reducing the impact of noise and extreme values.

F  ANALYSIS OF OVERFITTING IN CONCEPT LEARNING

Figure 10: Visualization of training dynamics of one run on CEBaB and IMDB-C datasets. We adopt the “early stop” strategy to avoid overfitting, where the model with the highest validation accuracy is selected and evaluated on the test set.

G  COMPUTATION COMPLEXITY AGAINST NUMBER OF EXPERTS

In Figure 11, we show an experiment to quantify the increase in computational complexity, as measured by FLOPs, with the increasing number of experts, in one the inference given a sequence of 128 tokens. The results, as shown in the attached figure, clearly indicate a linear increase in computational complexity with the number of experts. This is an expected outcome, as each additional expert adds a similar computational load to the model. However, it is crucial to note that the addition of each expert provides a significant improvement in model performance. We acknowledge that there is a trade-off between the increased computational overhead and the enhanced performance achieved. In practical applications, this trade-off would need to be carefully balanced based on the specific requirements and constraints of the task at hand. In future work, we plan to investigate techniques for optimizing the computational efficiency of our model, such as pruning less effective experts or implementing more efficient computation strategies. This will allow us to maintain or even enhance model performance while managing the computational costs more effectively.