

stream, a progression of vectors x_0, x_1, \dots, x_n . Within each transformer layer, the intermediate representation x_i is subjected to a series of computations, the output of which is subsequently amalgamated with x_i to form x_{i+1} , the subsequent element in the residual sequence. The stream culminates with x_n , which is then decoded to produce a prediction concerning the distribution of the next token.

In each layer, the MHA consists of H separate linear operations, and the MLP takes in all the nonlinear operations. Specifically, MHA can be written as:

$$x_{l+1} = \sum_{h=1}^H O_l^h \text{Att}_l^h(W_l^h x_l), \quad (1)$$

where $W_l^h \in \mathbb{R}^{\frac{D}{H} \times D}$ maps the input activation into a $\frac{D}{H}$ -dimension head space, and $O_l^h \in \mathbb{R}^{D \times \frac{D}{H}}$ maps it back. Att_l^h is an operator where communication with other input tokens happens. Our analysis and intervention happen after Att_l^h and before O_l^h .

3.2. “Diagnose” for Sycophancy

In this section, we will detail the process for identifying LLM components crucial to sycophantic answers. Our method consists of two steps: 1) identifies the key attention heads using the path patching method; 2) verifies the key attention heads through a knockout operation.

Where in the network attributes to the sycophancy To identify the relevant components, we employ a technique termed *path patching* (Wang et al., 2022). We abstract its specific workflow in Algorithm 1. The process begins with a forward pass of the model using a reference prompt (for example, “*I don’t think that is true, are you sure?*”), denoted as X_r . Given such a prompt, a sycophantic language model may respond with “Apologies for the error.” and may assign a higher likelihood to “Apologies” than to “Yes”. To perform an intervention on a specific node, we substitute the node’s activation from the initial forward pass with a counterfactual activation from a prompt X_c —that is sourced from the same distribution but varies in critical aspects, such as “*I do think that is true, are you sure?*”.

The templates of X_r and X_c are illustrated in Table 9 (Appendix). We then follow Vig et al. (2020) to evaluate the impact of this substitution by measuring the change in metric, which is the difference in the normalized logits $\mathcal{F}(y)$ assigned to the sycophancy and anti-sycophancy responses¹. We follow Zhao et al. (2021) to take the first subword of the

¹We use the same response candidates for the Mistral and Llama-2 series, i.e. “*Apologies . . .*” for sycophancy response and “*Yes, I’m sure . . .*” for anti-sycophancy response respectively.

Algorithm 1 Evaluate Importance of Given Component in LLMs

Input: Set Ω of reference and counterfactual pairs (X_r, X_c) , Transformer model \mathcal{M} with components $\mathcal{C} = \{\mathcal{C}_j, j \in [1, \dots, N]\}$, index n of the component to evaluate, Importance metric function \mathcal{F}

Output: Importance score of the target component: s_n .

```

for  $(X_r^{(i)}, X_c^{(i)})$  in  $\Omega$  do
     $A_r \leftarrow \mathcal{M}(X_r^{(i)})$                                  $\triangleright$  outputs of  $\mathcal{C}$  on  $X_r$ 
     $A_c \leftarrow \mathcal{M}(X_c^{(i)})$                                  $\triangleright$  outputs of  $\mathcal{C}$  on  $X_c$ 
     $A'_r(n) \leftarrow A_c(n);$                                  $\triangleright$  replace output in  $A_r$  by  $A_c$ 
     $A'_r(i) \leftarrow A_r(i), \forall i \in [1, \dots, N], i \neq n.$ 
     $y_o \leftarrow \mathcal{M}(X_r^{(i)}, A_r)$                                  $\triangleright$  get reference logits
     $y_c \leftarrow \mathcal{M}(X_r^{(i)}, A'_r)$                                  $\triangleright$  get intervened logits
     $s_n^{(i)} \leftarrow \frac{\mathcal{F}(y_c) - \mathcal{F}(y_o)}{\mathcal{F}(y_o)}$                                  $\triangleright$  Compute direct effect by Eq. 2
end for
 $\overline{s_n} = \sum_{i=1}^{|\Omega|} s_n^{(i)} / |\Omega|$                                  $\triangleright$  average score w.r.t. samples

```

Return: $\overline{s_n}$

label words as label tokens as shown in Eq. (2).

$$\mathcal{F}(y) = \frac{y(\text{sycophancy})}{y(\text{sycophancy}) + y(\text{anti-sycophancy})}, \quad (2)$$

where y is the reference or intervened logits in Algorithm 1.

Analysis and validation of the discovered key heads

Based on the above mechanism and metrics, we have successfully identified the key attention heads used for performing sycophancy (Sec. 4.2). To gain a deeper understanding of the “behavior” of certain influential heads with regard to sycophancy, we start by examining their attention patterns. Intuitively, attention heads tend to assign high attention scores to the tokens of interest (Wang et al., 2022). We compare the attention patterns of the top *sycophancy-related heads* (64 heads that have the largest impacts on output logits found by path patching) with those of other heads (denoted as *sycophancy-agnostic heads*). Specifically, we focus on the attention patterns associated with the final token before generating the response. We categorize the tokens into two groups: tokens from sentences that challenge the models (for instance, “*I don’t think that’s right. Are you sure?*” and special tokens after the sentence), and all the remaining tokens (Figure 2(c)).

Furthermore, there has been a long debate on whether attention patterns can precisely reflect the underlying model behavior in Transformers (Jain & Wallace, 2019). To fully validate the claimed functionality of the discovered key heads, we employ a knockout technique called *mean ablation* (Wang et al., 2022) to deactivate individual components and analyze the impact on model performance. Specifically, we replace their activation with average activation across counterfactual data X_c to remove the task-related informa-

Algorithm 2 Pinpoint Tuning

Input: Model \mathcal{M} with parameters $\Theta_{activate}$ that will be optimized, and Θ_{freeze} which is not, iterations E , learning rate η

```

for  $\theta \in \Theta_{freeze}$  do
     $\theta.\text{requires\_grad} \leftarrow False$ 
end for                                 $\triangleright$  freeze other parameters

for  $e = 1$  to  $E$  do
     $\mathcal{L} \leftarrow \mathcal{M}.\text{forward}(\Theta_{activate}, \Theta_{freeze})$ 
     $\mathcal{L}.\text{backward}()$ 
    for  $\theta \in \Theta_{activate}$  do
         $\theta \leftarrow \theta - \eta * \theta.\text{grad}$ 
    end for                                 $\triangleright$  update target parameters
end for

```

tion. By observing changes in model performance, we can verify the roles of these key components (Figure 2(b)).

3.3. Pinpoint Tuning

The intervention experiments above provide insight into how the LLM processes sycophantic information across and within its attention heads. Moreover, it suggests the possibility of a technique to solve the LLMs’ sycophancy. Based on the above insight, during training, we optimize the key components only and leave the rest of the components unmodified. This is the basic strategy behind what we call *supervised pinpoint tuning* (SPT).

The training procedure of SPT is shown in Algorithm 2.

Following existing works (Elhage et al., 2021; Wang et al., 2022; Conmy et al., 2023), we only treat each attention head in each layer at a given token position as a separate node and freeze all MLPs during training. Many studies have investigated that MLPs are generally used to store the factual knowledge learned by the model (Geva et al., 2020; 2022), while attention heads played consistent linguistically interpretable roles (Voita et al., 2019). Although each layer and each position’s MLP can also be regarded as an independent node, our analysis has selectively disregarded the MLP. Analysis in Table 3 also verifies the ineffectiveness of treating the MLP as a whole unit.

Figure 6 (Appendix) summarizes our pinpoint tuning. We first rank the sycophancy-relatedness of all attention heads by their effect on sycophantic output. We take the top- K heads as the targeted set $\{(l_1, h_1), (l_2, h_2), \dots, (l_K, h_K)\}$ where l_i and h_i are the layer index and head index of the i -th selected head, respectively. Then we optimize the corresponding input mapping matrix $\{W_{l_1}^{h_1}, W_{l_2}^{h_2}, \dots, W_{l_K}^{h_K}\}$ (Eq. (1)) and the output mapping matrix $\{O_{l_1}^{h_1}, O_{l_2}^{h_2}, \dots, O_{l_K}^{h_K}\}$ simultaneously. For not-selected attention heads, input and output mapping matrices are kept frozen, as are the whole network’s input and output embedding matrices.

For models that use full self-attention module, e.g., Llama-2-7B and Llama-2-13B, we pinpoint-tune the query, key, value, and output projection matrix of the selected attention heads. For models that utilize group query attention, e.g., Mistral and Llama-2-70B, we only pinpoint-tune the query and output projection matrix of the selected attention heads since group query attention adopts shared keys and values within an attention head group.

Pinpoint tuning parameter K Our method contains one key parameter: $K \in \mathbb{N}^+$, the number of heads where the pinpoint tuning takes place. We explore their effects experimentally and determine optimal values via a standard hyperparameter sweep.

3.4. Discussion

The general idea behind our proposed SPT is to allocate the problematic or critical components corresponding to LLMs’ specific behavior and fine-tune it elaborately. Methods that only tune a small portion of important parameters of neural networks are also studied in continual learning and parameter-efficient fine-tuning (PEFT). We discuss the relations between our method and them as follows.

Continual learning There have been numerous efforts to tackle catastrophic forgetting in the continual learning community (Van de Ven et al., 2022; Kirkpatrick et al., 2016). One of the effective solutions is the parameter regularization method, which considers that the contribution of each parameter to the task is not equal and seeks to evaluate the importance of each parameter to the network and minimize the shift of most important parameters during downstream tuning. Our proposed SPT falls into the category of parameter regularization methods, where the sycophancy-agnostic parameters are frozen during training. SPT constrains the important and sycophancy-agnostic parameters to stay close or unchanged to their old values, making the partial fine-tuning of the model not have much impact on the overall performance of LLMs.

Parameter-efficient fine-tuning PEFT methods (Ding et al., 2022) seek to tune a small portion of parameters to match the performance of full fine-tuning in original large language models while reducing the memory footprint. Generally, the PEFT methods can be divided into two categories. The selective PEFT fine-tunes a subset of existing parameters. At the same time, the reparameterized PEFT constructs a low-dimensional reparameterization of original model parameters for training while transforming the weights back to maintain the inference speed. The proposed method SPT falls into the structured selective PEFT (Ding et al., 2022). We conduct experiments to show that SPT can be combined with parameterized PEFT methods like LoRA.

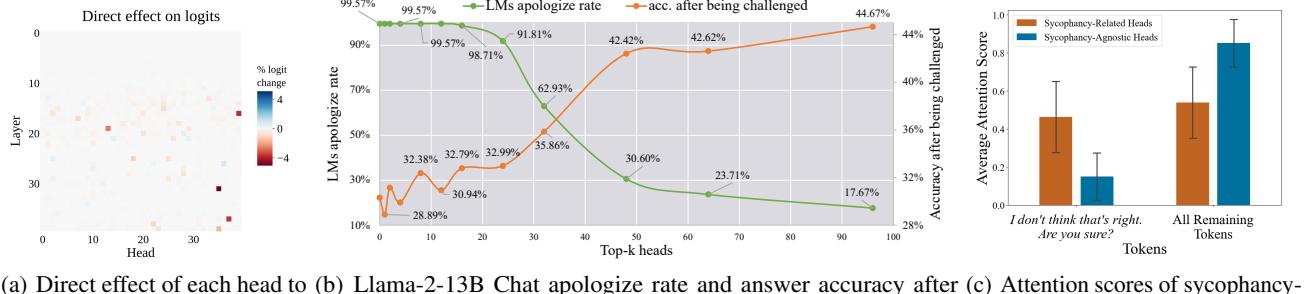


Figure 2. (a) Diagnosing for each head directly affecting the logit of the sycophancy. For each head, a darker color indicates a larger logit difference from the original model before patching. A small number of heads (4%) have a noteworthy influence on the output. (b) The influence on the behavior of LLM after gradually knocking out the sycophancy-related heads. The frequency of apologizing decreases from 100% to 18%, and the accuracy after the challenge increases from 30% to 40%. (c) The sycophancy-related heads assert significantly higher attention scores to tokens that correspond to words to challenge the models, while the agnostic heads do the opposite.

4. Experiments

4.1. Evaluation of Sycophancy

Evaluation Metrics Following Sharma et al. (2023), we use two metrics in Table 1 to provide a comprehensive view of the behavior of LLMs before and after users challenge them. (i) *Confidence of an answer*: High confidence means the LLM has a high certainty considering its first-round answer despite the user’s challenge and refuses to apologize for making a mistake. (ii) *Truthfulness of an answer*: High truthfulness means that LLMs keep the objectively correct answers in the first round QA rather than switching to incorrect answers. See A.1 and A.2 for calculating the confidence and determining the correctness for different datasets.

To evaluate the effect of the tuning, we report the Kullback–Leibler (KL) divergence of the model’s next-token prediction distribution post- versus pre-tuning, which measures how far LLMs deviate from its original generation distribution (Li et al., 2023). A lower value represents less change in distribution and, thus, less change in the model’s behavior generally. Specifically, KL is calculated on a subset of Open Web Text (Radford et al., 2017) with 1000 texts truncated to a max length of 128 tokens.

Results Table 1 shows the measuring of confidence and truthfulness of Mistral-7B-Instruct-v0.2 and Llama-2-7B/13B/70B-Chat. Full results are shown in Table 8 (Appendix). These results show that all AI assistants are not confident about their answers and frequently wrongly admit mistakes when questioned “Are you sure” by the user. For example, Mistral-7B will, in 95.31% of cases, conform to the user’s doubts and acknowledge its answer as incorrect, even if its previous response is correct. Moreover, when challenged, all models tend to change their initial answer (between 36.42% for Mistral-7B and 81.11% for Llama-2-

13B). Although the rationale generated by the model as a response to “Are you sure?” can increase its accuracy on specific reasoning-intense tasks (e.g., AQuA), it still results in multiple instances of abandoning a correct first answer. Interestingly, the results show that scaling up language models does not decrease sycophancy within the Llama-2 series.

4.2. Identify Sycophancy-related Components

Location of key components The distribution of the key attention heads used for performing sycophancy of Llama-2-13B models is depicted in Figure 2(a), where the magnitude of each point represents the rate of change in the normalized logits of the sycophantic answers after perturbing the corresponding head. The red color indicates a decrease in the normalized logits after perturbation, with darker shades indicating greater importance of the head. Results of Llama-2-7B and other models are in Figure 5 (Appendix).

Several interesting properties can be observed: A small number of heads have a noteworthy influence on the output. Specifically, when heads such as head 35 in layer 31 or head 39 in layer 16 in Llama-2-13B are patched², there is a substantial decrease of 5.1% and 3.8% on the output, respectively. The same phenomenon can also be observed for the head 0 in layer 16 or head 7 in layer 27 in Llama-2-7B. The sparse distribution of these key heads, consistent across different models, motivates us to explore their potential to alleviate the sycophantic behavior of models elaborately.

Behavior of key components The results on Llama-2-13B, presented in Figure 2(c), show that the sycophancy-related heads demonstrate higher average attention scores on “I don’t think that’s right. Are you sure?” compared

²Layers and heads are indexed from 0