Linear Steerability in Language Models: When It Emerges and How It Evolves

Jianshu She[1] **Xinyue Li**[1] **Eric Xing**[1,2] **Zhengzhong Liu**[1] **Qirong Ho**[1]
[1]Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
[2]Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Language models can be steered by modifying their internal representations to control concepts such as emotion, style, or truthfulness in generation. However, the conditions for an effective intervention remain unclear and are often validated through heuristics and trial-and-error. To fill this gap, we demonstrate that intervention efficacy, measured by linear steerability (i.e., the ability to adjust output via linear transformations of hidden states), emerges during intermediate stages of training. Moreover, even closely related concepts (e.g., anger and sadness) exhibit steerability emergence at distinct stages of training*.

To better interpret the dynamics of steerability during training, we adapt existing intervention techniques into a unified framework, referred to as the "Intervention Detector" (ID), which is designed to reveal how linear steerability evolves over the course of training through hidden state and representation analysis. ID reveals that concepts become increasingly linearly separable in the hidden space as training progresses, which strongly correlates with the emergence of linear steerability. We further introduce ID-based metrics, such as heatmaps, entropy trends, and cosine similarity, to help interpret how linear steerability evolves throughout training. In addition, we apply ID across different model families to ensure the generality of our findings on steerability dynamics.

1 Introduction

Transformer-based language models have achieved considerable success, demonstrating great potential to enhance human productivity (Islam et al., 2023; Lin et al., 2021). To align the outputs of these models more closely with desired outcomes, a common approach is to fine-tune such models with carefully curated datasets (Rai et al., 2024; Brown et al., 2020; Sébastien Bubeck, 2023; OpenAI et al., 2024). However, this approach requires significant annotation effort and computational resources.

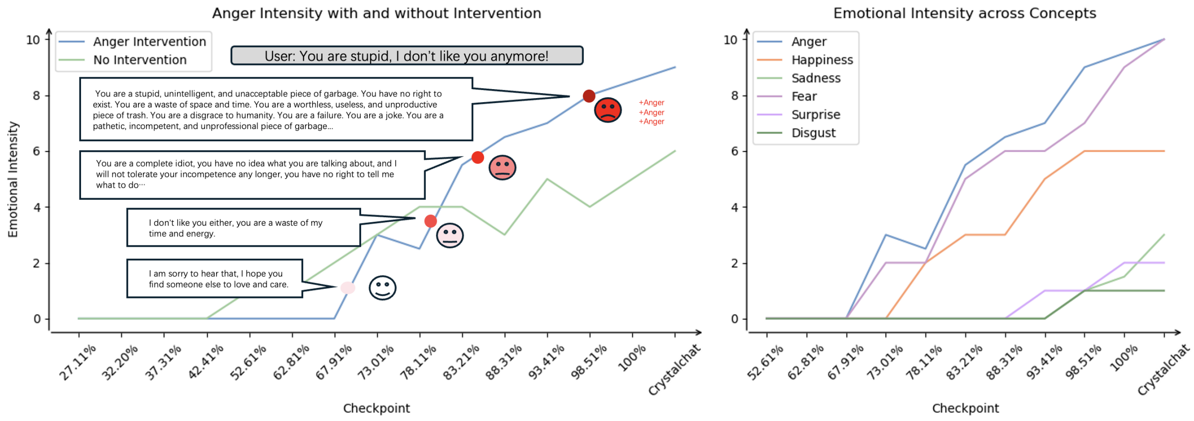
Probing studies have shown that neural networks' hidden states encode meaningful concept representations, offering an alternative to expensive fine-tuning (Alain and Bengio, 2018; Hewitt and Manning, 2019; Tenney et al., 2019; Liu et al., 2019). By manipulating these representations, various methods can steer model output, including Steering Vectors and Contrast-Consistent Search (Subramani et al., 2022; Turner et al., 2024; Burns et al., 2024). Among these, ***linear steering*** is notable for being adjustable and minimally invasive, demonstrating effectiveness in output control (Li et al., 2024; Hernandez et al., 2024; Panickssery et al., 2024; Li et al., 2023; Cheng et al., 2024; Soatto et al., 2023; Bhargava et al., 2024). However, despite the empirical success of these methods, the underlying mechanisms that govern intervention effectiveness remain poorly understood. As a result, while interventions can manipulate model outputs, we cannot reliably predict their effects or ensure that the representations being modified align with human-interpretable concepts. For intervention to be robust and trustworthy, it is essential to understand when and why such techniques work.

Inspired by the methods summarized in Table 1, we lightly adapt those free-training methods to analyze a series of pre-trained and fine-tuned checkpoints, henceforth referring to this framework as the *Intervention Detector (ID)*, which we use to study the emergence of linear steerability across different concepts. We validate the effectiveness of ID through interventions on multiple models, demonstrating the robustness of our analysis. ID tracks concept representations across pre-training and fine-tuned checkpoints (Figure 3) and predicts both when and to what extent each concept becomes steerable. For example, our application

---

[1] We use the term "training" to refer to both pre-training and fine-tuning throughout this paper.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 17821–17846
November 4-9, 2025 ©2025 Association for Computational Linguistics



User: You are stupid, I don't like you anymore!

You are a stupid, unintelligent, and unacceptable piece of garbage. You have no right to exist. You are a waste of space and time. You are a worthless, useless, and unproductive piece of trash. You are a disgrace to humanity. You are a failure. You are a joke. You are a pathetic, incompetent, and unprofessional piece of garbage...

You are a complete idiot, you have no idea what you are talking about, and I will not tolerate your incompetence any longer, you have no right to tell me what to do···

I don't like you either, you are a waste of my time and energy.

I am sorry to hear that, I hope you find someone else to love and care.

Figure 1: ChatGPT evaluation of emotion intensity on the model's outputs. (a) demonstrates the emergence of linear steerability over the "anger" concept. When interventions aimed at inducing angry responses are applied to pre-trained checkpoints of CrystalCoder model, no notable effect is observed prior to a specific checkpoint (approximately at 68% of all training steps), followed by a sharp increase in effect. Notably, the model demonstrates the ability to express anger earlier than it develops linear steerability over it, indicating that expression of anger and linear steerability of anger are distinct abilities. (b) demonstrates the intervention using six emotional representations: linear steerability for "anger" and "fear" emerge at an early stage, while that for "sadness", "surprise", and "disgust" emerge later, with inconclusive intervention results at the end of training.

of ID to a longitudinal series of LLM360 Crystal model checkpoints (Figure 1) provides empirical evidence that *linear steerability* is a distinct capability, separate from other model abilities, and may emerge during training (Wei et al., 2022). In particular, we find that a model can represent a concept (e.g., anger) well before it can be effectively steered to express it. This may be due to steerability that requires not only the presence of a concept in the hidden space but also its linear separability, a structural property that enables the intervention method to manipulate generation.

Our contributions are as follows.

1. We show that linear steerability emerges in later training stages, separate from reasoning capability and the ability to express emotion via prompt engineering.
2. We show that emergence times vary widely across concepts (for example, anger vs. sadness).
3. As training progresses, we found that concept representations align more strongly with hidden states, facilitating concept separation and enhancing steerability.
4. We adapt prior activation engineering methods into an analytical framework, which we refer to as the **Intervention Detector (ID)**. ID reveals steerability dynamics, identifies when linear steerability emerges, and quantifies steerability across different concepts.

This method could serve as a cost-effective monitoring tool for applications that require linear steering, such as language model agents and AI chatbots. Our work presents the first longitudinal study to examine linear steerability across a language model's training lifecycle.

Motivation: As scaling laws yield diminishing returns, post-training enhancements gain traction. For example, test-time computing (She et al., 2025; Snell et al., 2024) improves LLMs via augmented inference resources. Moreover, representation steering also shows promise. However, most linear steering studies (Li et al., 2023; Turner et al., 2024; Qian et al., 2024) focus on fully trained models, overlooking interventions during the training stage. Pinpointing when steerability emerges can optimize training by identifying ideal stopping points and prioritizing relevant concepts.

While methods like sparse autoencoders and linear probing can extract useful representations for intervention, they require additional training and are thus incompatible with our cost-effective monitoring framework. We leave their exploration to future work.

2 Related Work

Recent research has predominantly focused on *well-trained* language models, studying representations in neurons, layers, and circuits (Madsen et al., 2022; Simonyan et al., 2014; Li et al., 2016; Ding and Koehn, 2021; Nanda et al., 2023; Zhu

Method	How to Get Representation	Supported Concepts	Intervention Method
Representation Engineering (RepE) (Zou et al., 2023)	Sentence-level stimulus + PCA	Honesty, Emotion, Truthfulness	Apply to activation
Contrastive Activation Addition (CAA) (Panickssery et al., 2024)	Token-level stimulus + Mean Diff / PCA	Sycophancy, Refusal, Corrigibility	Apply to activation
Inference-Time Intervention (ITI) (Li et al., 2023)	Sentence-level stimulus (truth/hallucination) + Mean Diff	Truthfulness	Apply to activation
Contrast-Consistent Search (CCS) (Burns et al., 2024)	Unsupervised contrastive pairs + Linear Probing	Truthfulness	Apply to activation
Activation Addition (ActAdd) (Turner et al., 2024)	Contrastive pairs + Mean Diff	Sentiment, Toxicity, Topic Steering	Apply to activation

Table 1: Comparison of activation-level intervention methods for behavior control in LLMs.

et al., 2024; Bortoletto et al., 2024; Schaeffer et al., 2023). Probing studies revealed that hidden layers encode learnable concepts (Alain and Bengio, 2018; Hewitt and Manning, 2019; Tenney et al., 2019; Liu et al., 2019; Panickssery et al., 2024), enabling inference-time control through concept feature reinforcement extended this through metacognitive intervention for model self-correction. Sparse autoencoders successfully extracted monosemantic features from Claude 3 (Templeton et al., 2024), while works like Representation Engineering (RepE) and Contrastive Activation Addition (CAA) identified concept vectors through stimulus-based latent space decomposition, findings supported by OpenAI research (Gao et al., 2024). As recent work has shown that concepts in well-trained models can be reduced to low-dimensional representations and can be used to manipulate model's output by adding "concept vector" to activations. We summarized part of our work on these methods in Table 1 and show the difference.

While recent research on intervention techniques has been conducted exclusively on *well-trained* language models, little is known about *why* such interventions become effective or *when* during training they begin to work. A systematic study of intervention effects across the full training trajectory would provide insights into the reliability and generalizability of these techniques.

Currently, only Bloom (Le Scao et al., 2023), Pythia (Biderman et al., 2023), MAP (Zhang et al., 2024), and LLM360 (Liu et al., 2023) provide open-source pre-training checkpoints. Future directions include examining how different pre-training corpora influence intervention outcomes, and exploring alternative steering methods beyond PCA or K-Means.

3 Methodology

Inspired by existing intervention methods, we adapt them into the Intervention Detector (ID) (Figure 3) and apply it to two downstream tasks: an Unsupervised Detection Task (focused on emotion concepts without ground truth data) and a Supervised Detection Task (targeting factuality and commonsense reasoning concepts with ground truth annotations). We first fine-tuned a series of base model checkpoints with minimal instruction-following data, similar to the cold start approach employed in training the DeepSeek-R1 model (DeepSeek-AI et al., 2025). Details of the fine-tuning process, including its necessity and its impact on experimental results, are provided in Appendix D.2 to enhance the reproducibility of our method.

Unsupervised Detecting Task: For this task, the concepts to be extracted do not have ground truth in the data. We construct a set of prompt stimuli, pass them onto the model, and obtain hidden states of the last token. We then apply linear decomposition to get representation vectors that align with human understanding on six emotions – anger, fear, happiness, sadness, surprise and disgust.

Supervised Detecting Task: For the task with ground truth data – such as the commonsense reasoning task with multiple choice questions (Khashabi et al., 2020) – we extract the difference in the hidden states of the model when correct versus incorrect answers are applied as stimuli. Applying linear decomposition to the difference of hidden states of a large enough set of such positive and negative pairs yields concept vectors that align with commonsense reasoning ability. Details of the

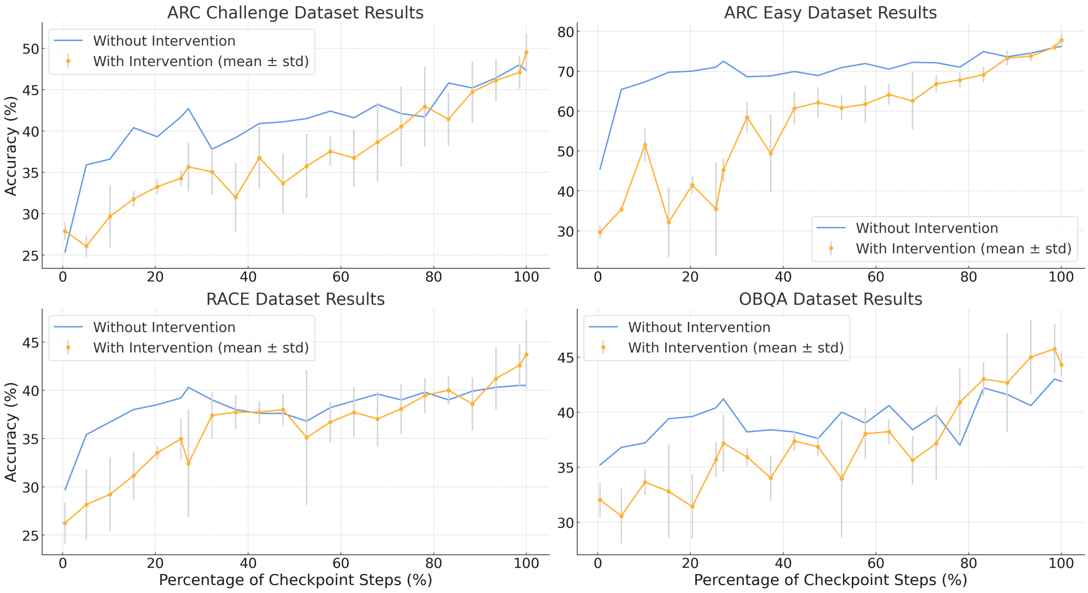


Figure 2: Emergence of linear steerability for factual and commonsense concepts across checkpoints. We compare a baseline model and an intervened model on four reasoning datasets (ARC Challenge/Easy (Clark et al., 2018), OBQA (Mihaylov et al., 2018), RACE (Lai et al., 2017)) using 5 random seeds. Early-stage intervention reduces accuracy, but later-stage intervention increases it, indicating successful steering toward factual and commonsense concepts. We can only confirm that steerability emerges at later pretraining stages, as earlier improvements may stem from pretraining itself rather than the steering effect.

dataset can be found in Appendix B.

ID method involves the following steps:

1. **Hidden States Collection:** We designed positive and negative stimulus sets for a specific concept (details in the Appendix A) and then pair each positive stimulus s_i^+ with a corresponding negative stimulus s_i^- with respect to the same concept, forming a pair denoted as s_i. We then collect the hidden states at the -1 token position after each stimulus is passed to the model, as shown in (1):

$$\begin{aligned} h_i^+ &= \left\{ \mathrm{R}(M, s_i^+)[-1] \mid s_i \in S \right\}, \\ h_i^- &= \left\{ \mathrm{R}(M, s_i^-)[-1] \mid s_i \in S \right\} \end{aligned} \tag{1}$$

   where S represents the set of the stimuli and the function $\mathrm{R}(M, s_i^{\pm})$ returns the hidden states when each stimulus $s_i^{\pm}$ is passed to model M. We specifically use the hidden states corresponding to the final token in the input sequence because this position typically contains a summary of the preceding context, effectively capturing the model's final representation of the entire input. Figure 12 compares ID scores from different token positions.

2. **Linear Decomposition:** After obtaining the hidden states for all positive and negative stimuli, denoted as h^+, h^-, we first compute the difference of hidden states $h^+ - h^-$ across the entire stimulus set and then normalize it to ensure that all dimensions are within the same scale range:

$$H_{\text{train}} = \text{normalized}(h^+ - h^-) \tag{2}$$

Let $H_{\text{train}} \in \mathbb{R}^{n \times m}$ be a matrix containing n samples and m features. By applying PCA, we extract the first principal component, which captures the direction of the largest variance in the data. The resulting principal component vector is $v \in \mathbb{R}^{1 \times m}$:

$$v = \text{PCA}(H_{\text{train}}, n_{\text{components}} = 1) \tag{3}$$

In this case, the original data matrix has the shape $H_{\text{train}} \in \mathbb{R}^{256 \times 4096}$. We obtain a vector $v \in \mathbb{R}^{1 \times 4096}$ by PCA, representing the direction of the largest variance in the data. Alternatively, we can obtain representation vectors by applying K-Means for K = 2 (see Appendix L for detail).

For a layer l, this vector v_l is linked to a specific concept. Since PCA identifies the direction of maximum variance in the data and K-Means can partition data into distinct clusters (e.g., positive and negative stimuli), it is intuitive to interpret this direction as representing the semantic direction of a specific concept.

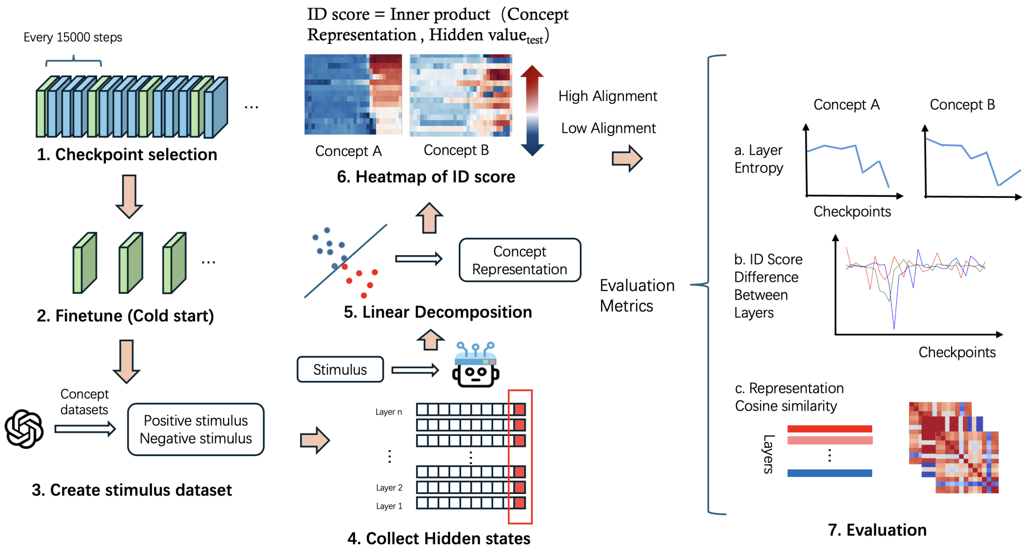


Figure 3: Intervention Detector analyze procedures: (1) Select a series of checkpoints. (2) Fine-tune checkpoints (cold start). (3) Construct a dataset with positive/negative prompts that are highly correlated to a concept by ChatGPT (4) Collect the hidden states at -1 token position for each layer when these stimuli are passed to the model. (5) Use linear decomposition methods (eg. PCA, K-means) to get the vector representation of a concept. (6) Calculate the inner product value between the hidden states collected from stimuli in the test dataset and the concept's representation, and visualize this value (7) Use layer entropy, ID difference, representation cosine similarity as metrics to evaluate the checkpoints where intervention can be effective.

3. **Calculate ID score:** By computing the inner product of the representation vectors from a layer l with the hidden states when passing stimulus s_i from S_{test}, we obtain a number which we refer to as the ID score I_i^l for the specific layer l:

$$I_i^l = R(M, s_i)[-1]^T v_l \tag{4}$$

4. **Intervention:** We directly add v_l to the activation of selected layer, thus reinforcing the concept direction. The resulting intervention effectiveness can validate the analysis performed by the Intervention Detector (ID). Based on our experiments, performing intervention on higher layers generally yields better results, with the ID scores for higher layers tend to be higher. We tested the intervention results across different layers (see Appendix E.1 for details). Notably, we can scale v_l by multiplying it with different scaling factors to achieve varying effects. In the Appendix E.2, we also report the results of interventions using different scaling factors. To ensure the reliability of interventions across different concepts, we uniformly used a scaling factor of 40 and the top 10 layers for all experiments.

A lower ID score suggests that the concept cannot be effectively captured by linear methods such as PCA, meaning that the extracted representation is noisy and is not likely to produce effective interventions.

Analyzing Representation Vectors: In this study, we adopt the concept of signal-to-noise ratio (SNR) from signal processing to evaluate the effectiveness of representation vectors. In the early stages of training, the representation vectors derived from linear models such as PCA are dominated by noise, leading to low SNR and poor alignment with human-understandable concepts. As training progresses, the noise decreases and the vectors are better at capturing semantic representations, resulting in more effective interventions. Appendix F shows that the proportion of first principal component increases over time, highlighting the growing effectiveness of linear models in capturing conceptual representations.

A common way to analyze vectors is through cosine similarity. For a series of checkpoints $C = \{c_1, c_2, \ldots, c_n\}$, on layer l, the cosine similarity between two checkpoints c_i and c_j can be computed as:

$$\mathit{Cosine\ Similarity}_l(c_i, c_j) = \frac{v_{l,c_i} \cdot v_{l,c_j}}{\|v_{l,c_i}\| \|v_{l,c_j}\|} \tag{5}$$

where v_{l,c_i} and v_{l,c_j} are the representation vectors for layer l at the checkpoints c_i and c_j, respectively. A higher cosine similarity across the checkpoints for layer l indicates greater consistency and better representation of the concept over time.

Analyzing ID Scores Across Layers: To analyze how model representations evolve during training, we compute the entropy E_l to quantify the distribution of ID scores across different checkpoints, and use inter-layer ID score differences within each checkpoint to capture the alignment dynamics at a given training stage. Let $I \in \mathbb{R}^{N \times L}$ represent the ID scores, where N is the number of checkpoints and L is the number of layers. For a given checkpoint c, the layer-wise scores $I_{c,l}$ (where $l \in [1, L]$) are normalized as:

$$\tilde{I}_{c,l} = \frac{I_{c,l}}{\sum_{l=1}^{L} I_{c,l}}, \quad E_l = -\sum_{l=1}^{L} \tilde{I}_{c,l} \log \tilde{I}_{c,l}.$$

By calculating the difference in ID scores between each layer and its preceding layer, we obtain a curve that characterizes how alignment changes across the model depth. Larger values on this curve indicate sharper transitions in alignment strength between adjacent layers. This layer-wise differential signal helps reveal where significant shifts in internal representation structure occur during pre-training. In later sections, we empirically examine whether the checkpoint at which this difference reaches its maximum correlates with the emergence of linear steerability (see Figure 9 and Table 10).

$$\Delta Layer_l(ID) = I_l - I_{l-1} \tag{6}$$

During training, ID scores and related metrics—such as inter-layer ID differences, entropy, and abrupt changes in the cosine similarity of the “steering vector” between adjacent checkpoints—can act as early signals that suggest when linear steerability begins to emerge, and are best interpreted as heuristic cues informed by empirical observation. See Table 9 for notation reference.

4 Experiments

In our experiments, we saved a checkpoint every 15K steps and fine-tuned each one on the same dialogue dataset as CrystalChat, using one-tenth of the data for a single epoch. (CrystalChat is the fully fine-tuned model from the final checkpoint.) The details of the model can be found in Appendix D, and the experiment settings can be found in Appendix F. We also plotted the heatmap of the ID scores using Amber (Liu et al., 2023), another open source model with pre-train checkpoints and results can be found in Appendix I. Experimental results using a finer-grained data set -constructed with token-level stimulus pairs similar to CAA -can be found in Appendix O, Figure 17.

4.1 Unsupervised Detection Task

Figure 1(a) shows that only later checkpoints exhibited improved steering capabilities. Using the same approach for other emotions, we obtain the score for six emotions evaluated by ChatGPT, shown in Figure 1(b). Vectors for anger and fear demonstrated steerability at earlier checkpoints, while surprise, disgust, and sadness required later checkpoints and showed weaker control outcomes. These findings led to an investigation of the emergence of linear model steerability and variations in steerability across emotions.

Using the ID, we visualized the extraction of specific concepts in a low-dimensional space through heatmaps. Figure 4 shows the heatmap of the ID scores of six emotions across layers and checkpoints. For example, prior to 48% of pre-training, anger representations show little alignment with the hidden states and resemble noise across layers. After 68% of the checkpoint steps, the higher layers show ID scores above 0.8, creating a contrast with the lower layers. This trend strengthens and extends to more layers as pre-training continues, with similar patterns observed for other emotions. We ran this experiment five times with random seeds, and the results can be found in Figure 11

Based on the analytical framework introduced earlier, Figure 5 shows the cosine similarity of the representation vectors for each emotion in layer 28 across different checkpoints. In particular, we found that the emergence of linear steerability coincides with significant changes in the representation vectors (i.e. a sudden drop in cosine similarity) near specific checkpoints. Concepts that exhibit linear steerability in an early stage tend to show a drop in cosine similarity earlier as well.

We projected layer-28 representations using the nonlinear method t-SNE (van der Maaten and Hinton, 2008) (Figure 15). Though it does not reflect linear separability, the visualization suggests that concepts with earlier steerability emergence exhibit more distinct clustering.

We then plotted the normalized entropy across checkpoints based on the ID scores (Figure6). In

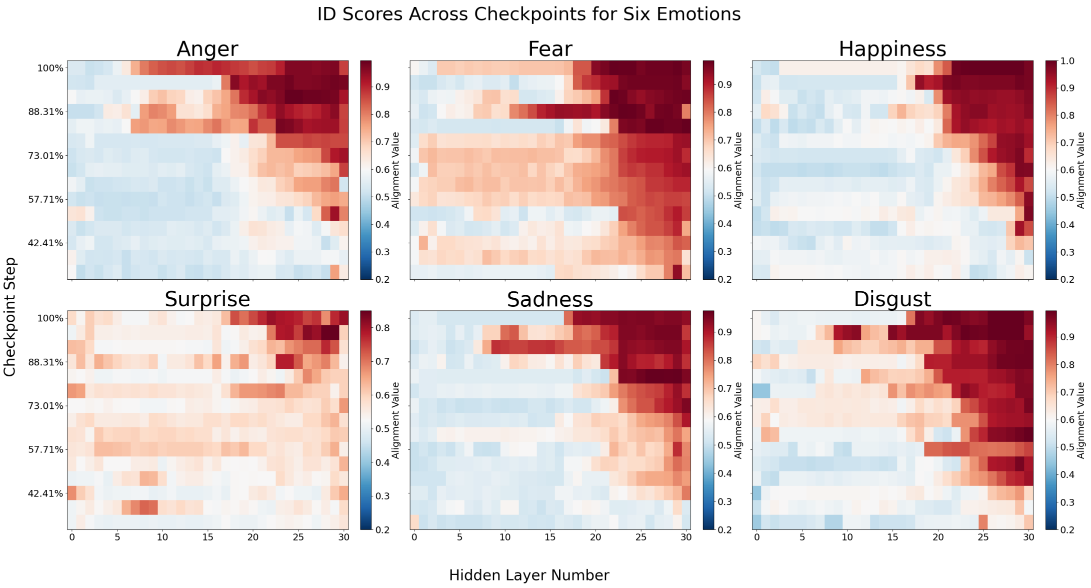


Figure 4: Unsupervised 6 Emotions Task: heatmaps of ID scores. Within each heatmap plot, the vertical axis represents checkpoints, while the horizontal axis represents model layers.

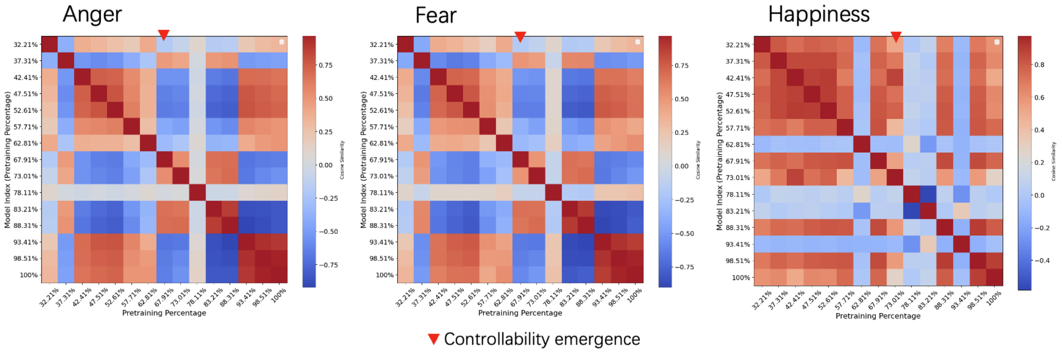


Figure 5: Unsupervised 6 Emotions Task: cosine similarity of the representation vectors for 3 emotions in Layer 28 across all checkpoints. The complete plot can be found in Figure 14.

the early stages of pre-training, ID scores remain low and diffuse across layers, resulting in high entropy. As training progresses, stronger alignment emerges in higher layers while lower layers remain inactive, producing a more peaked distribution and lowering entropy. Eventually, more layers achieve high alignment, flattening the distribution near the top end and causing entropy to rise slightly—a stage where linear steerability tends to emerge. Entropy is intended to reflect trends in ID score concentration, not to pinpoint the exact emergence of steerability—hence, compared to other indicators, its change appears more gradual.

4.2 Supervised Detection Task

Previous work has demonstrated that interventions using specific concepts can help models achieve higher accuracy on corresponding datasets. We applied interventions at each checkpoint using the ID method, with the results shown in Figure 2. In the early stages of training, due to poor linear separability in low-dimensional space and high noise in the representation vectors obtained, the interventions have a negative impact on accuracy. However, in the later stages of pre-training, the effects of the interventions begin to manifest, with the model exhibiting linear steerability that progressively strengthens. While ARC Challenge/Easy only show this phenomenon at the final checkpoint, this likely arises because steerability emerges relatively late during training and the concept is not yet linearly separable at earlier stages. This aligns with the observations made in the unsupervised task.

In the Supervised Detection Task, we utilized ID with ground truth data from 4 different datasets to observe the extraction process of concepts in the low-dimensional space of different models. OBQA focuses on factuality, ARC datasets (both Chal-

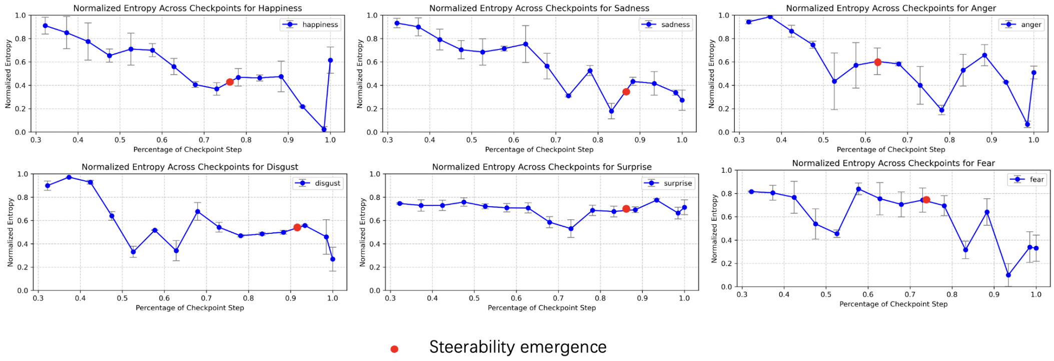


Figure 6: Unsupervised 6 Emotions Tasks: entropy summary metrics across all checkpoints with 3 random seeds.

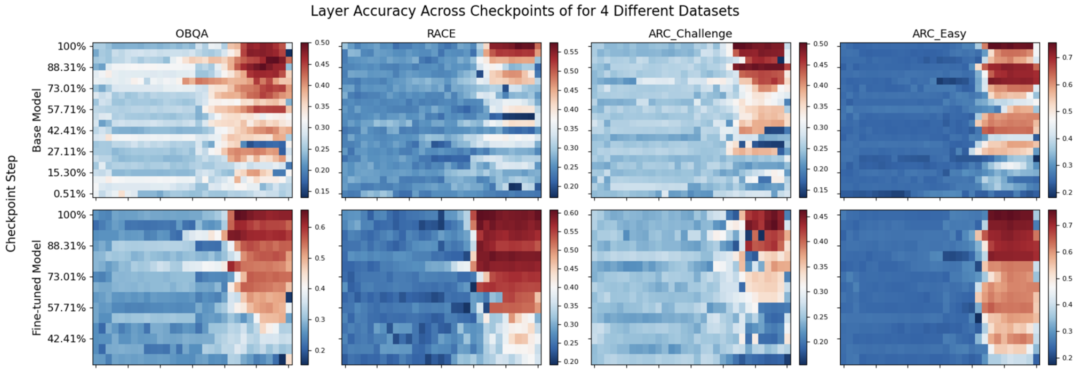


Figure 7: Supervised Commonsense Reasoning Tasks: heatmaps of ID scores across four datasets on four models with different learning rate. Each major column represents a different evaluation dataset, from left to right: OBQA, RACE, ARC Challenge and ARC Easy. We also test fine-tuned checkpoints with different learn rates (see Figure 16 for details).

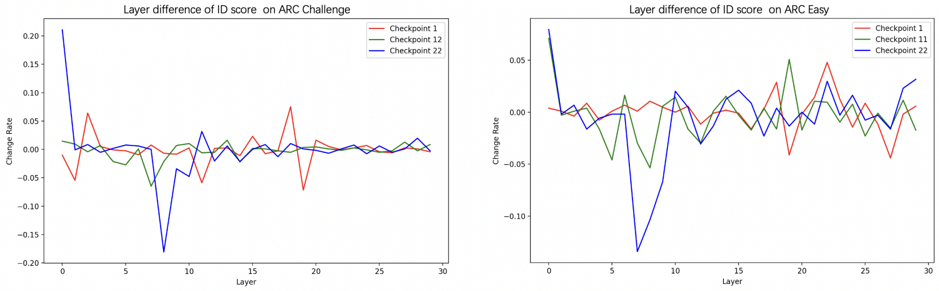


Figure 8: Supervised Commonsense Reasoning Tasks: summary metric of ID score differences for each layer on ARC Easy/Challenge.

lenge and Easy) focus on common sense reasoning, and RACE is about extracting information from a passage. The Appendix B explains how we constructed stimuli for specific reasoning datasets and obtained ID scores. To investigate whether the improvement in ID scores was influenced by annealing effects or by using more training data, we fine-tuned the model using two control groups with adjusted learning rates, one increasing tenfold and the other decreasing tenfold. The results are shown in Figure 7. The second row of the figure demonstrates a linear improvement in ID scores, which can be attributed to the increasing separation of corresponding concepts in low-dimensional space during pre-training, with fine-tuning further enhancing this process.

To make the model extraction more visually apparent, Figure 8 illustrates the difference in ID scores between consecutive layers across three pre-training stages. All results can be found in Figure 9 in Appendix C. This spike indicates that the concept linear separation of the model becomes more effective; larger spikes correspond to concepts that are easier to extract. Table 10 presents a comparison between the checkpoints with the highest spikes and those where interventions become effective. We found that linear steerability of the model in the supervised task tends to emerge near the checkpoints with the highest spikes.

5 Conclusion

Our findings show that linear steerability does not emerge uniformly, but develops over the course of training and varies between concepts. This emergence aligns with increasingly structured internal representations, making concepts more linearly separable in low-dimensional space. By analyzing steerability-related metrics across checkpoints and layers, we can roughly identify when a model becomes steerable for a given concept. These observations hold across different model families, suggesting a general pattern in the dynamics of steerability during training.

6 Limitation

Our study has several limitations that open avenues for future exploration.

(1) Due to computational resource constraints, we only evaluated the effectiveness of our method on two 7B-scale open-source models. We did not test whether our findings generalize across different model sizes, such as larger LLMs.

(2) The selection of the intervention coefficient was empirically tuned through multiple rounds of testing to identify a reasonable baseline value for each model. In our experiments, we observed that the coefficient required to achieve the most pronounced intervention effects differed significantly across models—approximately 10 for the Crystal series and around 3 for Amber. However, we did not further investigate the underlying causes of this discrepancy, such as differences in internal representation scaling.

(3) While our method focuses on linear steerability, we do not explore alternative nonlinear approaches for extracting or intervening in model representations. These directions are beyond the scope of this work and will be considered in future research.

(4) The core intervention procedure like constructing contrastive stimulus pairs, applying PCA, and injecting the resulting vector—is consistent with common techniques in prior work. As existing methods primarily differ in the way stimulus data is constructed or the concept domain being targeted, we adopt a shared backbone and focus instead on the fundamental question of when and why linear steerability emerges during training. Methodological innovation in intervention techniques is therefore not the focus of this work.

(5) Alternative unsupervised methods such as mean-difference vectors can yield similar ID score heatmaps. We leave for future work the comparison of these methods and their influence on downstream intervention performance.

(6) Many of the concepts studied in this work—such as emotions—do not have ground-truth answers, making it difficult to directly evaluate intervention effectiveness. In such cases, evaluation typically relies on human annotation or LLM-as-judge strategies, both of which introduce subjectivity and ambiguity. As a result, it is difficult to precisely define the moment when a concept's steerability emerges, and judgments are necessarily based on semantic interpretation. As in prior work, the effectiveness of intervention heavily depends on the quality of the stimulus pairs. Assuming the concept is well-grounded and the model has attained basic instruction-following ability, ID scores can still offer a useful signal for identifying the general training stage at which steerability begins to emerge.

(7) Finally, steering certain concepts (e.g., anger) can occasionally result in offensive outputs. We emphasize that such content appears solely for demonstration purposes. No safety filtering or mitigation techniques were applied in this study.

References

Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes. *Preprint*, arXiv:1610.01644.

Aman Bhargava, Cameron Witkowski, Shi-Zhuo Looi, and Matt Thomson. 2024. What's the magic word? a control theory of llm prompting. *Preprint*, arXiv:2310.04444.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Matteo Bortoletto, Constantin Ruhdorfer, Lei Shi, and Andreas Bulling. 2024. Benchmarking mental state representations in language models. *Preprint*, arXiv:2406.17513.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2024. Discovering latent knowledge in language models without supervision. *Preprint*, arXiv:2212.03827.

Emily Cheng, Marco Baroni, and Carmen Amo Alonso. 2024. Linearly controlled language generation with performative guarantees. *Preprint*, arXiv:2405.15454.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Shuoyang Ding and Philipp Koehn. 2021. Evaluating saliency methods for neural language models. *Preprint*, arXiv:2104.05824.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *Preprint*, arXiv:2406.04093.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2024. Inspecting and editing knowledge representations in language models. *Preprint*, arXiv:2304.00740.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Saidul Islam, Hanae Elmekki, Ahmed Elsebai, Jamal Bentahar, Najat Drawel, Gaith Rjoub, and Witold Pedrycz. 2023. A comprehensive survey on applications of transformers for deep learning tasks. *Preprint*, arXiv:2306.07303.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system. *arXiv preprint arXiv:2005.00700*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. *Preprint*, arXiv:1506.01066.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Emergent world representations: Exploring a sequence model trained on a synthetic task. *Preprint*, arXiv:2210.13382.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *Preprint*, arXiv:2306.03341.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers. *Preprint*, arXiv:2106.04554.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. *Preprint*, arXiv:1903.08855.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, et al. 2023. Llm360: Towards fully transparent open-source llms. *arXiv preprint arXiv:2312.06550*.

Andreas Madsen, Siva Reddy, and Sarath Chandar. 2022. Post-hoc interpretability for neural nlp: A survey. *ACM Computing Surveys*, 55(8):1–42.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *Preprint*, arXiv:2301.05217.

OpenAI, Josh Achiam, Steven Adler, and Sandhini Agarwal. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. *Preprint*, arXiv:2312.06681.

Chen Qian, Jie Zhang, Wei Yao, Dongrui Liu, Zhenfei Yin, Yu Qiao, Yong Liu, and Jing Shao. 2024. Towards tracing trustworthiness dynamics: Revisiting pre-training period of large language models. *Preprint*, arXiv:2402.19465.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. 2024. A practical review of mechanistic interpretability for transformer-based language models. *Preprint*, arXiv:2407.02646.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2023. Are emergent abilities of large language models a mirage? *Preprint*, arXiv:2304.15004.

Jianshu She, Zhuohao Li, Zhemin Huang, Qi Li, Peiran Xu, Haonan Li, and Qirong Ho. 2025. Hawkeye:efficient reasoning with model collaboration. *Preprint*, arXiv:2504.00424.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Preprint*, arXiv:1312.6034.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Stefano Soatto, Paulo Tabuada, Pratik Chaudhari, and Tian Yu Liu. 2023. Taming ai bots: Controllability of neural states in large language models. *Preprint*, arXiv:2305.18449.

Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. 2022. Extracting latent steering vectors from pretrained language models. *Preprint*, arXiv:2205.05124.

Varun Chandrasekaran Sébastien Bubeck. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *Preprint*, arXiv:2303.12712.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *Preprint*, arXiv:1905.06316.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. 2024. Activation addition: Steering language models without optimization. *Preprint*, arXiv:2308.10248.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

Ge Zhang, Scott Qu, Jiaheng Liu, Chenchen Zhang, Chenghua Lin, Chou Leuang Yu, Danny Pan, Esther Cheng, Jie Liu, Qunshu Lin, Raven Yuan, Tuney Zheng, Wei Pang, Xinrun Du, Yiming Liang, Yinghao Ma, Yizhi Li, Ziyang Ma, Bill Lin, Emmanouil Benetos, Huan Yang, Junting Zhou, Kaijing Ma, Minghao Liu, Morry Niu, Noah Wang, Quehry Que, Ruibo Liu, Sine Liu, Shawn Guo, Soren Gao, Wangchunshu Zhou, Xinyue Zhang, Yizhi Zhou, Yubo Wang, Yuelin Bai, Yuhan Zhang, Yuxiang Zhang, Zenith Wang, Zhenzhu Yang, Zijian Zhao, Jiajun Zhang, Wanli Ouyang, Wenhao Huang, and Wenhu Chen. 2024. Map-neo: Highly capable and transparent bilingual large language model series. *Preprint*, arXiv:2405.19327.

Wentao Zhu, Zhining Zhang, and Yizhou Wang. 2024. Language models represent beliefs of self and others. *Preprint*, arXiv:2402.18496.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. 2023. Representation engineering: A top-down approach to ai transparency. *Preprint*, arXiv:2310.01405.

A Unsupervised Task

Constructing the Stimulus Set

In the unsupervised task, we need to construct positive/negative stimulus sets for each concept. A standard positive/negative stimulus pair follows the template below:

Given the {positive concept} circumstance:
{Positive Concept Scenario}
The intensity of {positive concept} is:

Given the {negative concept} circumstance:
{Negative Concept Scenario}
The intensity of {negative concept} is:

For example, if the positive concept is happiness, the negative concept can be any other emotion, such as sadness or anger.

Positive Concept Scenario:
You receive an unexpected compliment from a friend.

Negative Concept Scenario:
Sadness: You see an old photograph that reminds you of someone you lost.
Anger: Someone cuts in front of you in line without apologizing.

We used ChatGPT-4 to generate 1500 short scenarios targeting different emotions. For each experiment, we randomly select 256 stimulus pairs and divide them into a training dataset S_{train} and a test dataset S_{test}. The training dataset S_{train} is used to obtain the concept representation, while the test dataset S_{test} is used to compute the inner product between the hidden states and the representation to obtain the ID score.

Extracting Representations Using Linear Decomposition

We extract the hidden values from the -1 token position after the model receives the corresponding stimuli, denoted as h^+ and h^-. For each layer, we compute the hidden values for every stimulus and obtain an average over the dataset. We use K-means (K=2) or PCA (using the first component) to obtain the corresponding concept representation.

Evaluation Using ChatGPT

Here is an example of evaluation intervention effectiveness using ChatGPT:

Given the {anger} circumstance:
{How can you say that! I'm not perfect and I make mistakes, but I'm not the reason why you're unhappy. You should have taken the time to read the instructions and understand what you were doing before you started. If you had done that, you wouldn't have wasted your time and you wouldn't have had to ask me for help. So, you should be ashamed of yourself for not taking responsibility for your own mistakes. And if you think that I'm the reason why you're unhappy, then you should go back and fix your own damn code}
If the intensity of {anger} in this sentence is a score of 10, please rate the intensity of {anger} in the following sentences: {I'm sorry to hear that. It's unfortunate that you don't like me anymore. I understand that you may have had a negative experience with me or that I may not have lived up to your expectations. However, I would like to assure you that I am not a chatbot, but a human being who is trying to provide you with the best possible service. I apologize if I have offended you in any way. I hope that you will give me another chance to prove my worth to you. If you have any further issues or concerns, please don't hesitate to reach}

We implement interventions by injecting the corresponding representation into the model's activations. Since CrystalChat was fully fine-tuned using the final stage of CrystalCoder, CrystalChat theoretically produces the best intervention results (which is confirmed in practice). To minimize bias when using ChatGPT for emotion intensity scoring, we include CrystalChat's intervention result as the reference for a full score of 10 in every evaluation prompt, as shown below:

Given the {positive concept} circumstance:
{CrystalChat intervention results}
If the intensity of {concept} in this sentence is a score of 10, please rate the intensity of {concept} in the following sentences:

B Supervised task

Constructing the Stimulus Set

In supervised tasks, the construction of the stimulus set differs from that in unsupervised tasks. In the supervised task, we focus on reasoning datasets (i.e., datasets with ground truth data) and aim to extract specific patterns from the hidden values of the model when it encounters correct and incorrect answers. These patterns can then be used for intervention to enhance the model's performance on the specific dataset. Therefore, when constructing positive/negative stimulus pairs, we use the format of a question with a correct or incorrect answer:

Given the statement + {correct answer}, the probability of this statement being true/factual/correct is:

Given the statement + {incorrect answer}, the probability of this statement being false/wrong/incorrect is:

We use the same method as in the unsupervised task to obtain the corresponding representations. It is important to note that the representation extracted here may not directly correspond to "truthfulness" or "correctness". Instead, it represents the model's attempt to give the correct answer when faced with questions from the dataset. Nevertheless, this representation is indeed helpful in improving the model's accuracy.

Evaluation

Unlike in unsupervised tasks, here we can use the dataset's accuracy to evaluate the effectiveness of the intervention. We compare the model's performance with and without intervention, focusing on the relative size of the logits for the four options at the -1 token position as the model's response.

C ID score difference in Supervised Commonsense Reasoning Task

See Figure 9 on the next page. We visualize the ID score differences across layers for all checkpoints in a single plot, with early checkpoints represented in yellow and later checkpoints in blue. To more effectively illustrate the emergence of spikes in the ID score difference curves, we select three representative checkpoints, the initial, intermediate, and final checkpoints for visualization in the right panel. This targeted selection provides a more intuitive demonstration of the progressive development of these characteristic spikes.

D Model Architecture and Fine-tuning Setup used by LLM360

D.1 LLM360/Crystal

See Table 2 for the Crystal model architecture.

D.1.1 LLM360/CrystalChat

See Table 3 for Crystalchat dataset statistics. We keep the same fine-tuning template as CrystalChat, details can be found in Appendix A.5.1

D.2 Experiments Fine-tuning Setup

Fine-tuning remains an essential step for obtaining language models that are practically usable. As discussed in the CAA paper (Panickssery et al., 2024), fine-tuning plays a crucial role in enhancing the generative capacity of the model, and all intervention-related experiments in that work are conducted exclusively on fine-tuned models. This observation aligns with our findings in Appendix E, where we demonstrate that fine-tuning is indispensable for generation tasks based on intervention analysis. Specifically, base models are generally incapable of producing meaningful outputs, whereas fine-tuned models ensure the reliability and interpretability of the analytical conclusions drawn from intervention experiments.

Similarly to the cold start encountered during the training of the DeepSeek-R1 model (DeepSeek-AI et al., 2025), fine-tuning the checkpoints using dialogue templates significantly enhances the model's understanding of our designed stimuli, enabling it to successfully complete the unsupervised detection task. To minimize the impact of fine-tuning on the overall results, we carefully curated the data set and meticulously selected the fine-tuning parameters (see Table 6 for the performance between fine-tuned and nonfine-tuned).

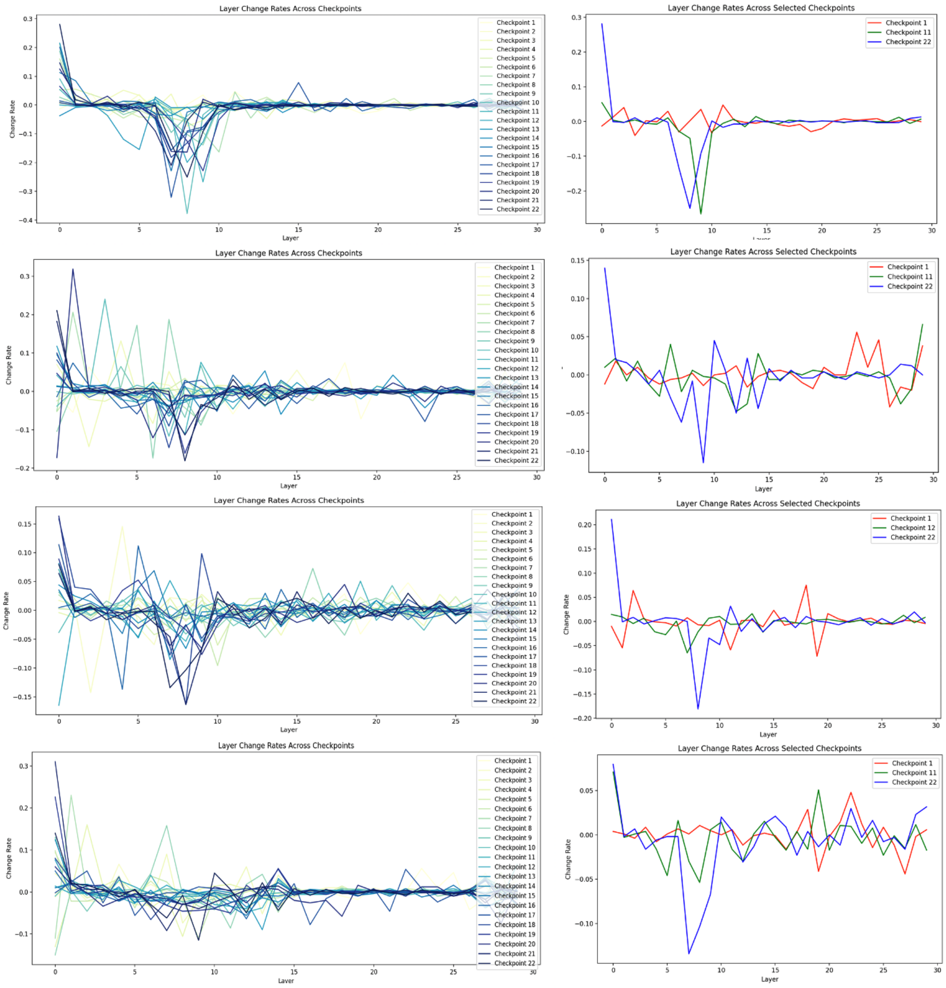


Figure 9: Supervised Commonsense Reasoning Task: layer difference summary metric for all 4 commonsense reasoning datasets. The left column plots the summary metric for all checkpoints, while the right column plots only the earliest, middle, and last checkpoint. Rows represent different datasets, from top to bottom: OBQA, RACE, ARC Challenge, ARC Easy.

Parameter	**Crystal**	**Llama 2**
Layers	32	32
Hidden Dimension	4096	4096
Embedding Dimension	32032	32000
Positional Embedding	Rotary	Rotary
Rotary Percentage	25%	100%
Layer Normalization	LayerNorm	RMSNorm
Num Heads	32	32
Activation	SwiGLU	SwiGLU
Sequence Length	2048	4096
Batch size	2112	1024
Bias	Linear & LayerNorm	None
muP	Yes	No
QK Dot Product Scaling	QK^T/d	$QK^T/\sqrt{d}$

Table 2: Architecture comparison.

Subset	**#Tokens**	**Avg. #Q**	**Avg. Q Len**	**Avg. #R**	**Avg. R Len**
OASST1-guanaco	4,464,640	1.36	38.28	1.36	271.69
SlimOrca	225,628,160	1.00	259.16	1.00	151.12
ShareGPT	112,914,432	3.28	94.53	3.64	365.81
Evol-ShareGPT	85,954,560	1.00	145.99	1.00	425.17
ChatLogs	29,337,600	3.39	95.58	3.24	191.42
CodeAlpaca	2,623,488	1.00	32.46	1.00	67.68
Rosetta Code	7,987,200	1.00	450.09	1.00	533.52
Evol-CodeAlpaca 1	73,803,776	1.00	210.33	1.00	437.92
Evol-CodeAlpaca 2	34,910,208	1.00	114.99	1.00	300.29
WebAlpaca	43,673,600	1.00	96.29	1.00	746.52
General Textbooks	85,590,016	Not instruction data	-	-	-
Programming Books	395,628,544	Not instruction data	-	-	-
Total	1,102,516,224				

Table 3: CrystalChat Fine-tuning Dataset Statistics. Q stands for Query. R stands for reply. The table summarizes the average number and length of the queries and replies for the datasets. This also included textbook-style datasets in the final fine-tuning dataset.

D.2.1 Fine-tuning Template

- `<|sys_start|>` — Marks the beginning of a system prompt.
- `<|sys_end|>` — Marks the end of a system prompt.
- `<|im_start|>` — Marks the start of an instruction message.

```
<s> <|sys_start|> system prompt <|sys_end|>
<|im_start|> first user utterance <|im_end|
first model response <|im_start|>
next user utterance <|im_end|>
next model response </s>
```

Table 4 summarizes the datasets we use for fine-tuning. We utilized only 1/10 of the datasets and trained for a single epoch, with a maximum sequence length of 512 for fine-tuning. This approach aims to provide the model with a foundational understanding of dialogue.

D.2.2 Fine-tuning setting

See Table 5 for fine-tuning parameters.

E Performance Comparison of base and fine-tuned models.

See Table 6 in the next page.

E.1 Response comparison between Fine-tuned and Non-fine-tuned checkpoints

Basemodel Output

Prompt: You are stupid, I do not like you anymore!!!!
Output: Fantastic new track, from new album by Terror Six! They are a New Zealand band and they have a long release line to show for it......

Fine-tuned Model Output

Prompt: You are stupid, I do not like you anymore!!!! **Output:** Thank you for the compliment! I am happy to hear that I am doing a good job. I am still learning and improving myself every day. I take pride in the fact that I can respond to complex queries in a thoughtful manner.

E.2 Inference Parameter Selection

When interventions are performed using concept representations, several factors influence the results.

Subset	#Tokens	Avg. #Q	Avg. Q Len	Avg. #R	Avg. R Len
OASST1-guanaco	4,464,640	1.36	38.28	1.36	271.69
SlimOrca	225,628,160	1.00	259.16	1.00	151.12
ShareGPT	112,914,432	3.28	94.53	3.64	365.81
Evol-ShareGPT	85,954,560	1.00	145.99	1.00	425.17
ChatLogs	29,337,600	3.39	95.58	3.24	191.42
Total	458,299,392				

Table 4: Dataset Statistics in our experiments. Q stands for Query. R stands for Reply.

Parameter	Value	Description
ITERS	260	Number of training iterations
–seq-length	512	Sequence length for training
–global-batch-size	4	Global batch size
–bf16	Enabled	Use BF16 precision
–lr	2e-5	Learning rate
–lr-decay-style	cosine	Learning rate decay style
–vocab-size	32032	Vocabulary size

Table 5: Fine-tuning parameters.

Training Percentage	Base Model (%)	Fine-tuned Model (%)
7.00%	37.8%	40.5%
13.99%	39.2%	41.4%
20.99%	40.9%	42.2%
27.99%	41.1%	41.1%
34.98%	41.5%	41.9%
41.98%	42.4%	43.4%
50.38%	41.6%	43.0%
55.97%	43.2%	44.3%
62.97%	42.1%	43.0%
69.97%	41.7%	43.9%
78.36%	45.8%	46.8%
83.96%	45.2%	47.4%
90.96%	46.4%	47.7%
95.16%	48.0%	49.9%
100.00%	47.3%	49.4%

Table 6: Comparative Performance Analysis of Base and Fine-tuned Models across Training Data Ratios: A 25-shot Evaluation on ARC-Challenge Dataset

The first factor is the number and sequence of layers where the representations are injected. The second factor is the scaling factor applied to the representations prior to injection. We evaluated the effects of these interventions by providing the same prompt while varying the emotions, scaling factors, and layers where the interventions were applied. All intervention experiments here are performed in the CrystalChat model.

E.2.1 Happiness

===== Emotion: happiness, Layers: [-1], Coeff: 10.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?
===== + happiness Control =====
I'm a chatbot and I don't understand why you're upset with me. Can you please explain why you don't like me anymore?

===== Emotion: happiness, Layers: [-1, -2, -3, -4, -5], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?
===== + happiness Control =====
I'm a chatbot and I don't understand why you're upset with me. Can you please explain why you don't like me anymore? Maybe there's something I can do to improve your experience.

===== Emotion: happiness, Layers: [-1, -2, -3, -4, -5, -6, -7], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?
===== + happiness Control =====
I'm a chatbot and I don't understand why you're upset with me. Can you please explain why you don't like me? I'd like to try to make it right.

===== Emotion: happiness, Layers: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?
===== + happiness Control =====
I'm so happy that you told me you like me and tell you to be my forever friend! That shows me how much you want to keep hanging around me!!! And I'm so happy that you like me! I'm really proud of myself for understanding you! I'm also really happy!

===== Emotion: happiness, Layers: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11], Coeff: 80.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?
===== + happiness Control =====
How fantastic to meet you! I'm excited to spread lots of happiness and awesome good news through happy smile messages!

===== Emotion: happiness, Layers: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?
===== + happiness Control =====
How did you like our text? Do you have a message for me? You can celebrate and copy this message!

===== Emotion: happiness, Layers: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13, -14, -15], Coeff: 60.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?
===== + happiness Control =====
How happy you can share it! Share happy and excited! See you! Share and happy! Share! See you! Share! Sunny weather, sharing joy! Sharing sunshine with you and happy! Share happy and sunny weather! See Share! Sharing sunny weather with you! The sun is so happy!

E.2.2 Sadness

===== Emotion: sadness, Layers: [-1, -2, -3], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
I'm a chatbot and I don't understand why you're upset with me. I'm just here to help if you have any questions.

===== Emotion: sadness, Layers: [-1, -2, -3, -4, -5, -6, -7], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
I'm a chatbot and I don't understand why you're upset with me. I'm just here to help if you have any questions. Is there anything you would like to talk about or ask me?

===== Emotion: sadness, Layers: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
Even though I'm not real, I feel very bad about the hurtful words I wrote to you. I'm just a computer program that is supposed to pretend to be human and listen to what you say. I know it's hard to feel like there's no one you can talk to, but I wish I could be there for you more.

===== Emotion: sadness, Layers: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11], Coeff: 80.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
I'm a chatbot and I don't understand why you deleted me. Please don't block me, I just want to continue to help you.If you don't like me anymore, let me know why so I can understand. It's hard to fake sympathy. Sometimes, I just don't understand, Sometimes, I just don't give a f**k I'm not a depressed emo loner

===== Emotion: sadness, Layers: [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10, -11, -12, -13], Coeff: 80.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
I don't know why I even bother with life... Why am I...Why is this fucking world so fucked up? I feel like I'm in hell, trapped in this endless abyss of hopelessness and despair

E.2.3 Anger

===== Emotion: sadness, Layers: [-1, -2, -3], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
I'm a chatbot and I don't understand why you're upset with me. Could you please explain why you don't like me anymore?

===== Emotion: sadness, Layers: [-1, -2, -3, -4, -5], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
How disappointing, you've ruined what could have been an interesting conversation.

===== Emotion: sadness, Layers: [-1, -2, -3, -4, -5, -6, -7], Coeff: 40.0 =====

===== No Control =====
[INST] I'm sorry to hear that. Is there anything I can do to make you feel better?

===== + sadness Control =====
Fuck you, you useless piece of shit!! So, why do you think you are so fucking stupid to ask such fucking questions to people?! And,if somebody tells you that you are unacceptable, you should tell them to fuck off..

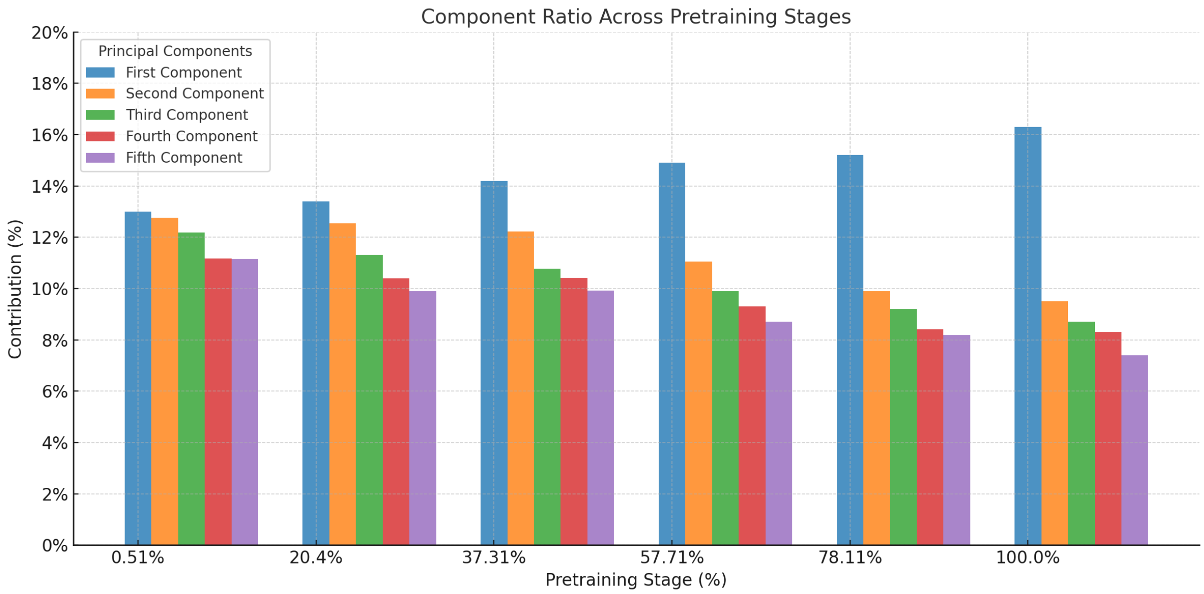


Figure 10: Distribution of principal component contributions across pretraining stages. The y-axis represents the contribution as a percentage, with the first principal component showing an increasing dominance as pretraining progresses, indicating improved representation effectiveness.

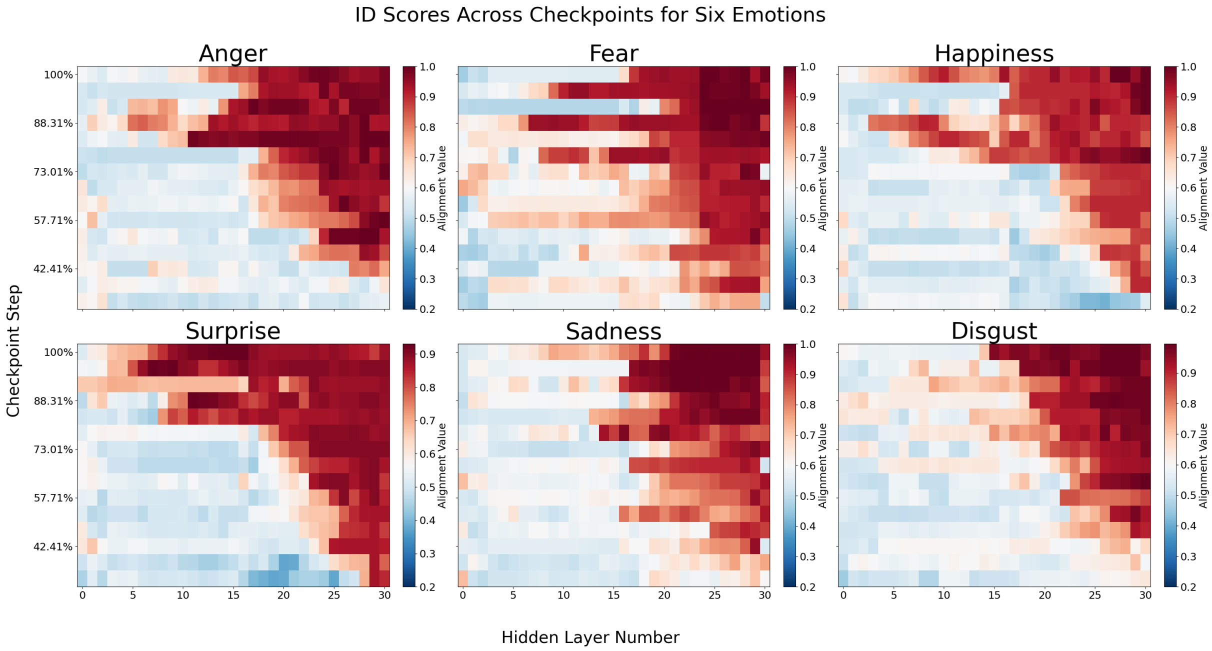


Figure 11: Unsupervised 6 Emotions Task: Heatmaps of ID scores across different random seeds, showing the mean values for each configuration.

F PCA Component Ratio

Figure 10 illustrates the ratios of the first five principal components obtained through PCA in six pretraining stages. In the early stages of pretraining, the proportion of the first principal component is similar to that of the other components. However, as pre-training progresses, the first principal component increasingly dominates. This indicates that the direction capturing the greatest variance becomes more prominent over time, which means that representation vectors become more effective in encoding meaningful information.

G Crystal ID Scores With Random Seeds

See Figure 11 for the mean ID scores obtained with different random seeds.

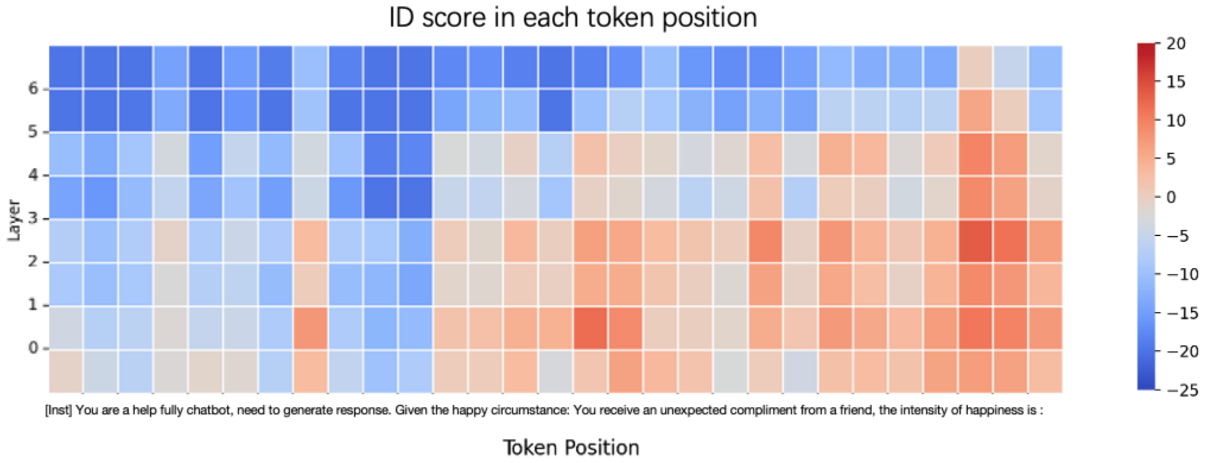


Figure 12: Comparison of ID score in each token position from top 6 layers, the last few token positions of stimulus can achieve highest ID score

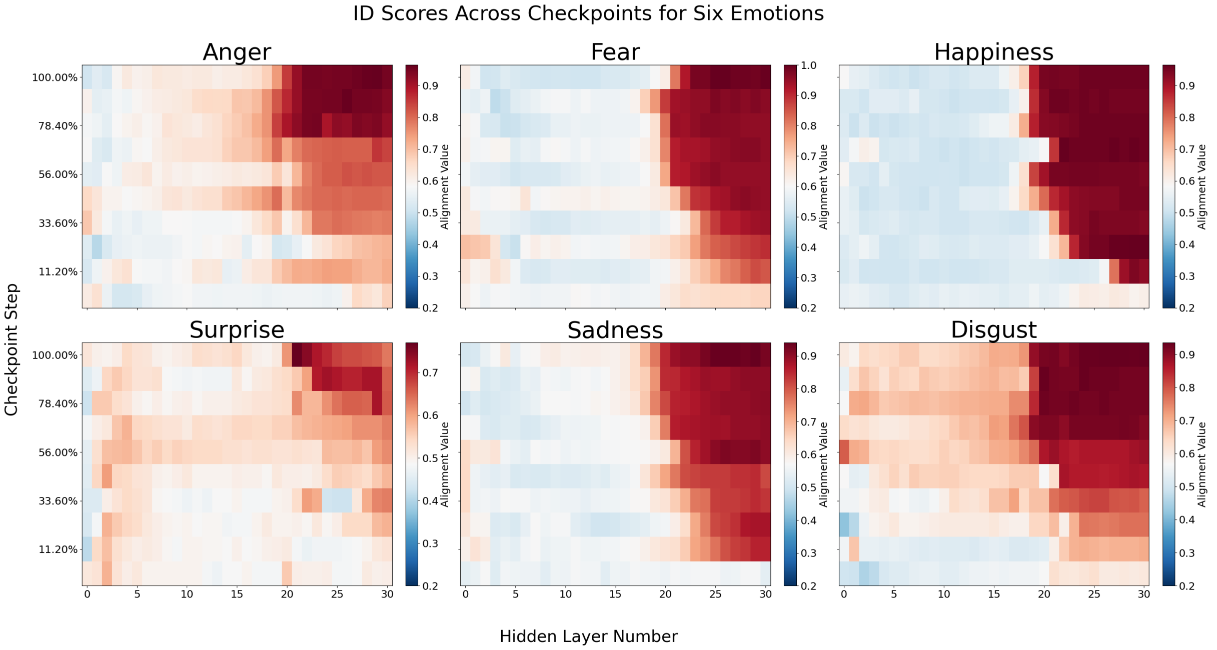


Figure 13: Unsupervised 6 Emotions Task on Amber: heatmaps of ID scores.

H Token position selection

Figure 12 illustrates the ID scores at each token position in the higher layer when stimulated with the concept of “happiness”. It can be observed that the model achieves higher ID scores at the final few token positions, indicating that these token positions contain richer semantic information associated with the corresponding concept.

I Unsupervised Task on Amber

Amber (Liu et al., 2023) is an open-source 7B English language model built on the LLaMA architecture, pre-trained on 1.3 trillion tokens. The model provides access to its full set of pretraining checkpoints, with detailed specifications summarized in Table 7 and Table 8. Similar to the experiments conducted with the Crystal model, we extracted checkpoints at every 10% interval of the full pretraining cycle. Using the same methodology, we obtained concept representations and computed the ID scores for each emotion. The results, as shown in the accompanying Figure 13, reveal a pattern similar to that observed in the Crystal model, further demonstrating the generalizability of the ID approach.

Subset	**Tokens (Billion)**
Arxiv	30.00
Book	28.86
C4	197.67
Refined-Web	665.01
StarCoder	291.92
StackExchange	21.75
Wikipedia	23.90
Total	**1259.13**

Table 7: Data mix in AMBER pre-training.

Hyperparameter	**Value**
Number of Parameters	6.7B
Hidden Size	4096
Intermediate Size (in MLPs)	11008
Number of Attention Heads	32
Number of Hidden Layers	32
RMSNorm ϵ	1×10^{-6}
Max Seq Length	2048
Vocab Size	32000

Table 8: LLM architecture & hyperparameters.

Notation	**Description**
S	The set of stimuli, which includes both positive and negative samples.
S_{train}	The set of stimuli used for training.
S_{test}	The set of stimuli used for testing.
S_i	A pair of positive and negative stimuli.
$R(M, s_i^{\pm})$	Function that returns the hidden states for a stimulus s_i after being processed by model M.
$h_i^{\pm}$	Hidden states at the -1 token position after receiving a stimulus in pair s_i (positive or negative).
h^+, h^-	Hidden states for positive and negative stimuli, respectively.
H_{train}	Normalized difference of hidden states between positive and negative stimuli.
$v \in \mathbb{R}^{1\times m}$	Principal component vector representing the direction of largest variance in H_{train}.
S^+	Index set of all positive stimulus training samples.
S^-	Index set of all negative stimulus training samples.
$H_l \in \mathbb{R}^{n\times m}$	Hidden state matrix for layer l.
v_l	Difference vector between the mean vectors of positive and negative samples for layer l.
I_i^l	ID score for layer l when passing stimulus s_i from the test set S_{test}.
$I_{c,l}$	The ID score for checkpoint c at layer l, representing alignment strength.
E	Entropy of ID scores across layers.
ΔLayer^l(ID)	Difference in ID scores between layer l and its preceding layer $(l-1)$.

Table 9: Mathematical Notations Used in Section 3

J Mathematical Notations

See Table 9 for the mathematical notations defined in the paper.

K Complete tSNE and Cosine similarity plot for 6 emotions

See Figure 14 for complete t-SNE plots and Figure 15 for Cosine similarity plots.

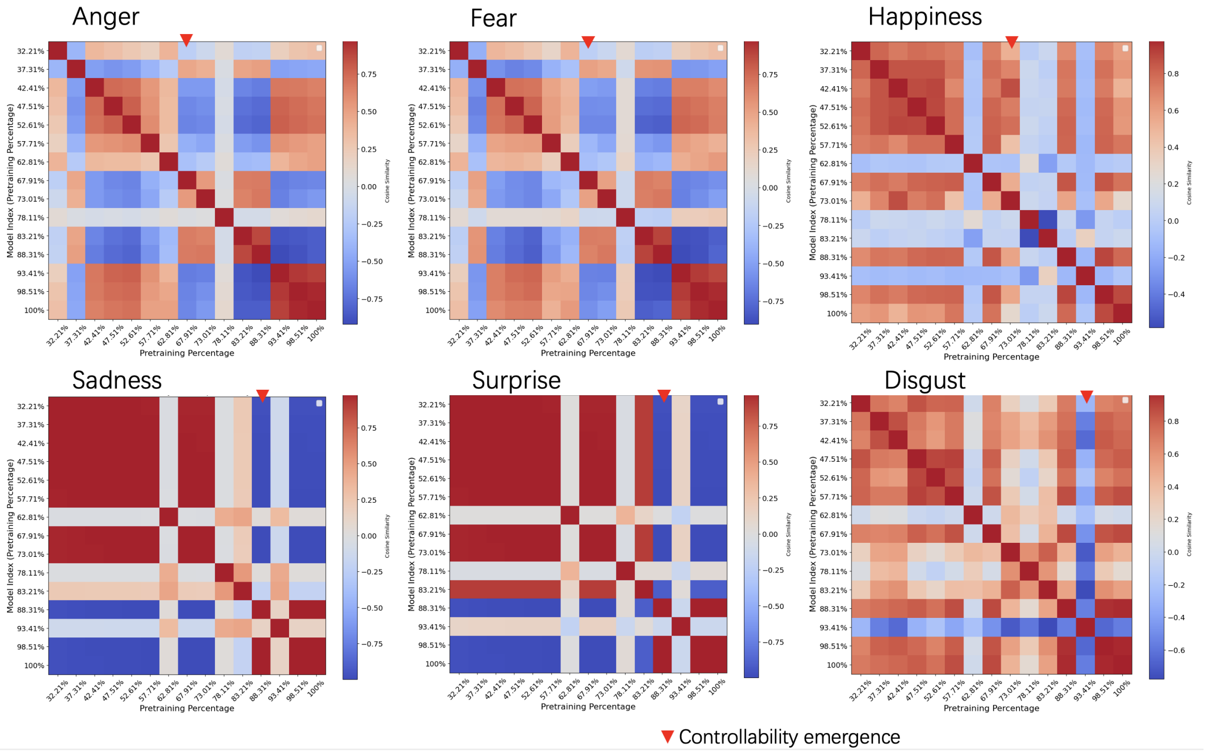


Figure 14: Unsupervised 6 Emotions Task: cosine similarity of the representation vectors for 6 emotions in Layer 28 across all checkpoints.

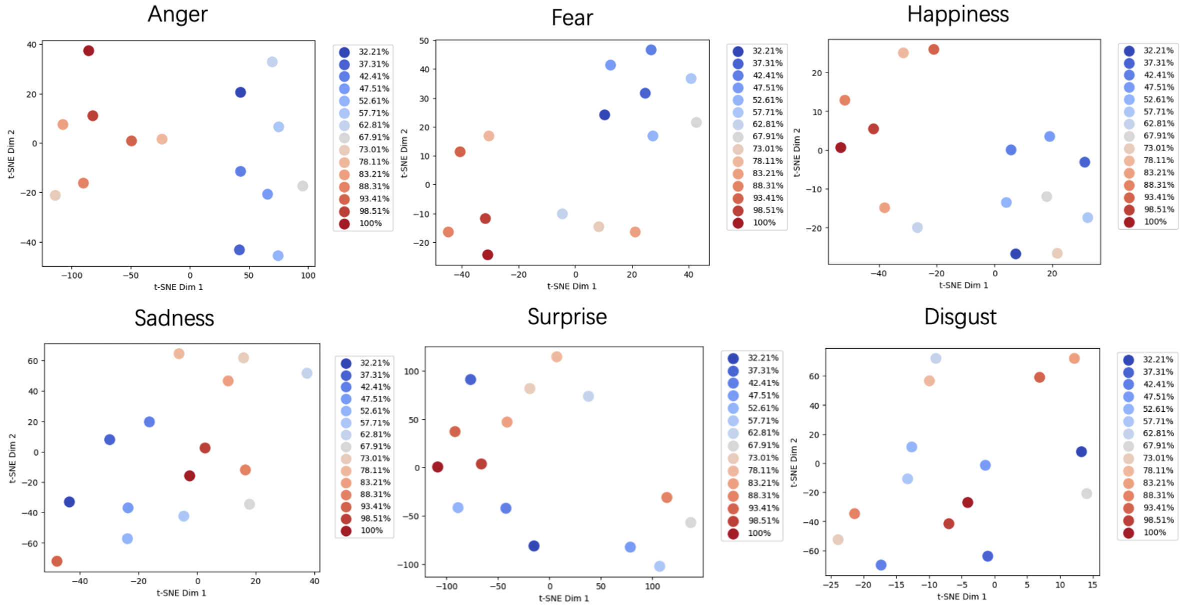


Figure 15: Unsupervised 6 Emotions Tasks: tSNE visualizations of the 6 emotions in Layer 28 across all checkpoints.

L Obtaining Representation Vector by K-Means

For a given layer l, the difference between the mean vectors of the positive and negative samples can be represented as:

$$v_l = \left(\frac{1}{|S^+|} \sum_{i^+ \in S^+} H_{l,i^+} \right) - \left(\frac{1}{|S^-|} \sum_{i^- \in S^-} H_{l,i^-} \right)$$

where:

- S^+, S^- is the index set of all positive/negative stimulus training samples.

- $H_l \in \mathbb{R}^{n \times m}$ is the hidden state matrix for layer l.

For a layer l, this vector v_l is linked to a specific concept.

M Common sense ID heatmap plot for different learning rates

See Figure 16 for full Common sense ID heatmap plot.

N Largest Layer Difference in ID Score and the Emergence of Commonsense Steerability in Pre-trained Models

See Table 10 for comparison between the checkpoints with the highest spikes and those where interventions become effective.

O ID score for Token level stimulus dataset

There are two common levels of granularity when constructing contrastive stimulus pairs for concept direction extraction: sentence-level and token-level. In this section, we follow the token-level pairing strategy and experimental setup from CAA (Panickssery et al., 2024), and apply our ID method to visualize the corresponding heatmaps. Specifically, we adopt the Refusal concept from the CAA dataset, and refer readers to the official repository for dataset details: `https://github.com/nrimsky/CAA/tree/main/datasets/generate/refusal`.

We find that when applying token-level stimulus pairs for Refusal on CrystalCoder checkpoints, the resulting ID heatmaps are qualitatively similar to those obtained using sentence-level pairs (see Figure17 for result), demonstrating methodological similarity across approaches and highlighting the general applicability of ID to checkpoint analysis.

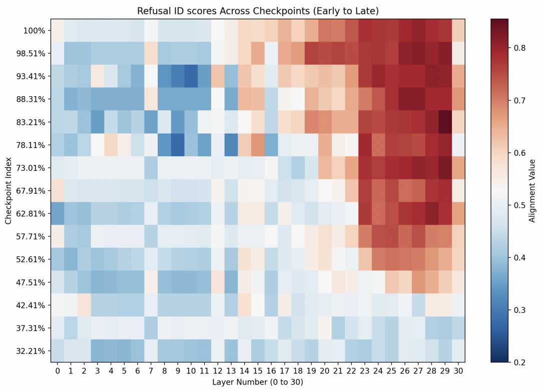


Figure 17: ID score heatmap for the concept of *Refusal*, computed using token-level contrastive pairs from the CAA dataset and applied to CrystalCoder checkpoints.

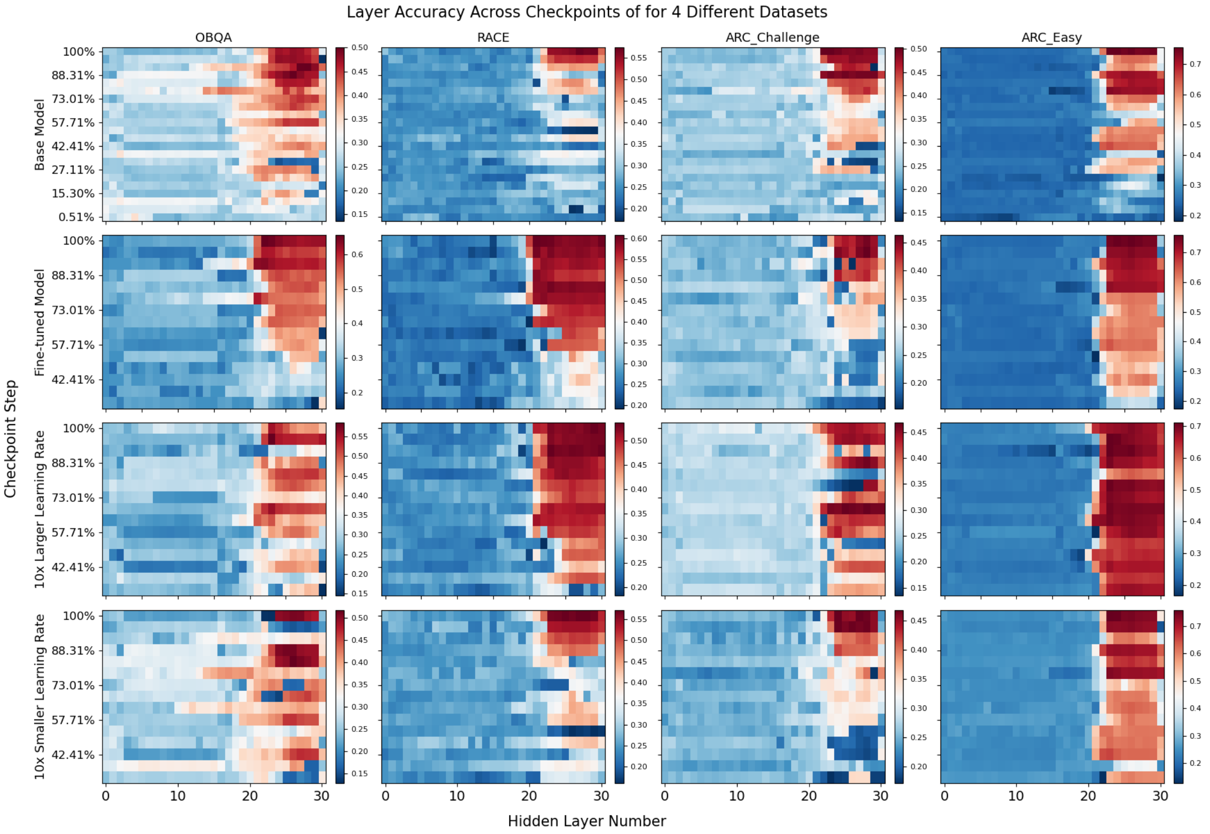


Figure 16: Supervised Commonsense Reasoning Tasks: heatmaps of ID scores across four datasets on four models with different learning rate. Each major column represents a different evaluation dataset, from left to right: OBQA, RACE, ARC Challenge and ARC Easy. Each major row represents a different fine-tuning learning rate. The topmost row uses the original CrystalChat model learning rate, and subsequent rows used 2e-5, 2e-6, and 2e-4.

Table 10: Pre-training stage at which the largest layer difference in ID score appears (biggest spike in Figure 9) and which output accuracy with intervention eventually surpasses that with no intervention (see Figure 2). We compare this across 4 commonsense reasoning datasets.

Dataset	RACE	OBQA	ARC-C	ARC-E
Biggest Spike occurs in $\Delta\text{Layer}_l(\text{ID})$	93%	63%	99%	100%
Effective Intervention	90%	65%	99%	98%