Decoupling Internal Representational Changes and Causal Importance in Fine-Tuned Large Language Models

Lingfang Li[†] **Procheta Sen**[†] **Shubham Das**[♣] **Danushka Bollegala**[†◇]
University of Liverpool, United Kingdom[†] IIEST, Shibpur, India[♣] Amazon[◇]
{L.Li85,procheta.sen,danushka}@liverpool.ac.uk[†]
2026itm016.shubham@students.iiests.ac.in[♣]

Abstract

Fine-tuning has emerged as a widely adopted approach for adapting Large Language Models (LLMs) to a variety of downstream tasks. However, how it reshapes their internal mechanisms remains poorly understood. To address this, we investigate how fine-tuning alters internal representations in LLMs, including attention patterns and layer-wise activations, and examine whether these changes are linked to task-relevant components identified by Edge Attribution Patching (EAP) (e.g., attention heads and logit-level activations) that drive task performance. We find that EAP-identified components are concentrated within specific layers, indicating a degree of functional localisation in how models internalise task-specific behaviour. Notably, the distribution of these components across layers is largely uncorrelated with the layers undergoing the most substantial representational changes during fine-tuning. Furthermore, we observe that overlap in EAP-identified components across tasks does not translate into cross-task performance transfer if the tasks are different in nature (e.g., classification vs. generative tasks). More specifically, fine-tuning on one task can lead to a degradation of performance on another when the two tasks exhibit a high degree of overlap in their EAP-identified components. Our code is available here. [1]

1 Introduction

Fine-Tuning (FT) has been widely used across diverse domains, including high-stake domains like biomedical (Wu et al., 2024; Lee et al., 2019; Liu et al., 2024; Luo et al., 2022) and legal applications (Chalkidis et al., 2020; Colombo et al., 2024; Tewari, 2024; Shu et al., 2024), (Jain et al., 2024; Rudman et al., 2023). However, there remains a limited understanding of how model-level representational changes relate to the underlying EAP-identified components (e.g., embedding nodes, attention heads, MLPs, and logit nodes) that are primarily responsible for task performance. It also remains unclear whether shared EAP-identified components across tasks drive cross-task performance transfer, a question that is crucial for developing efficient and principled FT strategies.

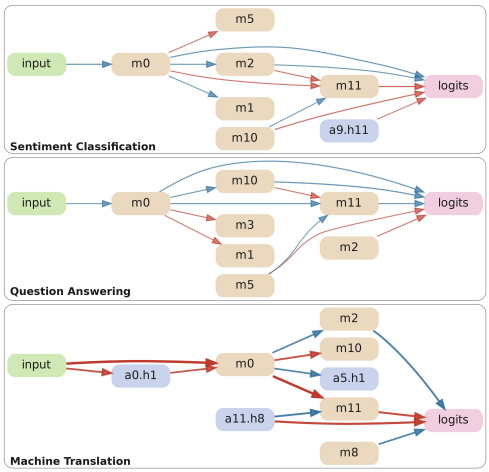


Figure 1: Examples of task-specific EAP-identified components (e.g., attention heads, MLPs) for sentiment classification (top), question answering (middle), and machine translation (bottom). Notations are described in Appendix A.

To address the above-mentioned issue, we study layerwise attention patterns and representation-level changes before and after FT. The reason for studying layerwise representations is to understand how task-level information encoded in every layer changes before and after FT. We used a causal attribution-based approach from mechanistic interpretability to find the causally relevant components in the fine-tuned model. Our primary objective is to investigate whether the layer undergoing the maximum representational change is also similar to the layer having the highest contribution to the EAP-identified components responsible for the per-

[1] https://github.com/LingfangLi/repr-change-vs-causal-importance

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 1767–1791
November 6-10, 2026 ©2026 Association for Computational Linguistics

formance of the task.

We investigate FT across three different Natural Language Processing (NLP) tasks (a) Sentiment Classification (label prediction), (b) Machine Translation (sequence generation), and (c) Question Answering (span extraction). To increase the generalisability of our findings, for each task, we select two datasets, resulting in six different datasets in total. Our experiments are conducted on GPT-2 Small (124M), Llama-3.2-1B, Qwen2-0.5B, Llama-2-7B. Figure 1 provides an illustrative example of EAP-identified components for three tasks (i.e., sentiment classification (SST-2), question answering (SQuAD), and machine translation (KDE4)), visualising a subset of the top-ranked components identified by our analysis. It can be observed that different kinds of tasks have similar and different causally relevant components. For example, MLP5 is present in both sentiment and question answering task, and the first attention head in the fifth layer (a5h1) is present only in machine translation. Broadly, our contributions can be summarised as follows.

- We perform a comprehensive causal component analysis rather than focusing only on the attention heads or feedforward layer similar to Yu and Ananiadou in three different NLP tasks (§ 5.2). We perform further analysis to quantify the overlap between known functional components, such as induction heads, and the identified causally relevant components in our research scope. In addition, we investigate the relative contribution of different component types (e.g., MLPs, attention heads, and logit nodes) among the top causally relevant components for each task (§ 6).

- We show that the layers undergoing the largest changes in attention patterns and representations during FT are not the layers that contain the most EAP-identified components for task performance (§ 5.1 and § 5.2).

- In contrast, for models fine-tuned on different tasks, we observe no performance gains despite the presence of overlap of EAP-identified components. This suggests that differences in task nature (e.g., classification vs. generation) restrict the effective reuse of EAP-identified components across tasks (§ 5.3).

2 Related Work

Mechanistic Studies of Fine-Tuning. Existing work has explored several techniques to understand the internal mechanisms of FT techniques. Prior studies have examined parameter, feature, and representation-level dynamics. Mukherjee et al. (2025) showed that FT can match full-update performance while modifying only sparse parameter subsets, Xu et al. (2025b) traced feature evolution across different phases of FT, Jain et al. (2024) used pruning and probing on procedurally defined synthetic tasks and argued that FT acts as a lightweight wrapper over pre-trained capabilities. Ren and Sutherland (2025) proposed a decomposition framework tracing how individual training examples shape predictions. Recent circuit-based work further studies FT through mechanistic interpretability. Wang et al. (2025) tracked circuits across FT checkpoints on mathematical tasks and found that FT primarily reorganises circuit edges rather than simply adding or removing nodes, motivating a circuit-aware LoRA method for FT. Prakash et al. (2024) showed that fine-tuning can preserve and strengthen existing task mechanisms rather than construct new ones, using entity tracking as a case study. Our work asks a different question: whether the components responsible for task performance are also those that undergo the most substantial changes during fine-tuning, and evaluates this across multiple task families. These works characterise what changes during FT, but they do not relate model-level representational change to the EAP-identified components responsible for task performance, which is the gap our work addresses.

Component and Circuit-Level Analyses of LLM Internals. A complementary line of work focused on localisation of internal components for different types of model behaviours. Prior studies identified task-relevant neurones in feed-forward networks (Geva et al., 2021; Wang et al., 2022; Xu et al., 2025a; Pan et al., 2024), used probing to investigate how task information is distributed across layers (Belinkov, 2022), or recover task-relevant circuits through attribution patching and automated circuit discovery (Syed et al., 2024; Hanna et al., 2024; Conmy et al., 2023). While these methods provide evidence that specific neurones, heads, layers, or edges contribute to model behaviour, many analyses focus on isolated component types (e.g., only on attention heads or only on feed-forward network) or on pre-trained models. Leng and Xiong

(2025) explored different task-specific neurones, specifically focusing on feed-forward layer networks to develop an efficient FT approach. Panigrahi et al. (2023) showed that a small, localized subset of model parameters can be sufficient to capture task-specific skills through parameter grafting. In contrast to the above-mentioned line of work, we use EAP to assign importance scores to edges throughout the model and select the top-k task-relevant edges across three different types of tasks. We then derive the corresponding sets of model components from these selected edges and examine whether their layer-wise causal importance aligns with the layers that change most during fine-tuning.

3 Methodology

Here, we describe the frameworks used to analyse model-level internal changes and identify EAP-identified components.

3.1 Quantifying Model-Level Changes

We focus on attention patterns and representation-level changes because they capture two complementary aspects of model behaviour under FT. Attention patterns provide insights into how the model routes information across tokens, reflecting dependency structure and contextual reasoning (Vaswani et al., 2017; Clark et al., 2019; Voita et al., 2019; Michel et al., 2019). In contrast, representation-level analyses capture how FT alters the geometry of the model's internal feature space, which encodes semantic and syntactic information (Ethayarajh, 2019; Hewitt and Manning, 2019; Kovaleva et al., 2019). Such changes reflect shifts in the model's learned abstractions beyond individual components.

Measuring Attention Pattern Changes. We first compare attention patterns before and after FT on each head across all the layers in an LLM. Given a fixed layer ℓ and a head h, for the k-th test instance, tokenised into a sequence of length d_k, let $A_{\ell,h}^{\mathrm{PT},k}, A_{\ell,h}^{\mathrm{FT},k} \in [0,1]^{d_k \times d_k}$ denote the corresponding attention pattern in the pre-trained and fine-tuned model, respectively. Each row in $A_{\ell,h,i,:}$ is a softmax output and thus a probability distribution over the d_k key positions ($A_{\ell,h,i,j} = 0$ for $j > i$ under causal masking), making the Kullback-Leibler (KL) divergence well defined between corresponding rows. Below, we measure the change magnitude of head h in layer ℓ by averaging the row-wise KL divergence over all d_k query positions, and then over n test instances as follows:

$$\Delta A_{\ell,h} = \frac{1}{n}\sum_{k=1}^{n}\frac{1}{d_k}\sum_{i=1}^{d_k}\mathrm{KL}\Big(A_{\ell,h,i}^{\mathrm{PT},k} \,\|\, A_{\ell,h,i}^{\mathrm{FT},k}\Big) \tag{1}$$

Taking the pre-trained pattern as the reference, Equation 1 measures how much FT shifts head (ℓ, h). We then average over the H heads, $\Delta A_\ell = \frac{1}{H}\sum_{h=1}^{H}\Delta A_{\ell,h}$, to obtain the average change in attention pattern per layer.

Measuring Representation Level Changes. To estimate whether task-relevant information remains encoded within different layers of fine-tuned LLMs in a way similar to that of the pretrained model, similar to Geva et al. (2023) and Jiang et al. (2024), we adopt a probing framework using logit lens (nostalgebraist, 2020). Specifically, for each test sample s and layer ℓ, we map the intermediate hidden state representation to the vocabulary space using the same output prediction head used for the final-layer representation, obtaining a per-layer prediction $\hat{y}_\ell^{(s)}$ for every sample $s \in \mathcal{S}$. The probing procedure is identical across tasks, what differs is the metric $\mathcal{M}$ used to compare $\hat{y}_\ell^{(s)}$ against the ground truth y_s.

$$\mathcal{M}_\ell \;=\; \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\mathcal{M}\big(\hat{y}_\ell^{(s)},\, y_s\big). \tag{2}$$

Equation 2 describes the mathematical formulation of the metric $\mathcal{M}_\ell$ used to measure the effectiveness of the representation obtained from a layer ℓ for an LLM for a task. We instantiate $\mathcal{M}_\ell$ as binary classification accuracy for sentiment classification, BERTScore (Zhang et al., 2020) for machine translation, and SQuAD F1 (Rajpurkar et al., 2016) for question answering. A higher $\mathcal{M}_\ell$ indicates that the representation at layer ℓ encodes more sufficient information to estimate the correct output, allowing us to compare how task-aligned information is distributed across layers in the base and fine-tuned models.

3.2 Quantifying Causal Relevance

Although the model can undergo multiple changes during FT, there may be certain components which are potentially causally responsible for the performance of a model in a task. To understand this, similar to Wu et al. (2023), we used the EAP technique (Syed et al., 2024) to estimate the important task-related components from an LLM.

Broadly speaking, EAP identifies the importance of each edge in an LLM, where an edge $E_{u\to v}$ exists between a source node u and a destination node v (e.g., outputs of previous layers and inputs of subsequent components). An edge essentially consists of two components while doing analysis related to causal relevance, we consider the union of all the components corresponding to top-k edges identified by EAP. EAP assigns a score to each edge. The saliency score $S_{u\to v}$ of an edge $E_{u\to v}$ is defined as the expected cumulative causal contribution to the task metric $\mathcal{L}$. To estimate the saliency score of an edge, we essentially corrupt an edge and observe the change in the model output. A first-order Taylor expansion is used to approximate the effect of corruption. Equation 3 gives the mathematical formulation of the saliency score for an input x.

$$S_{u\to v} = |E_{x\sim\mathcal{D}}\left[\langle \nabla_{\mathbf{a}_v}\mathcal{L}(x), \Delta\mathbf{a}_{v,x}\rangle\right]| \quad (3)$$

where $\Delta\mathbf{a}_{v,x}$ is the change in activation due to applying corrupted input to node u. Mathematically,

$$\Delta\mathbf{a}_{v,x} = \mathbf{a}_v(x) - \mathbf{a}_v(x') \quad (4)$$

In this formulation, $\mathbf{a}_v(x)$ denotes the activations generated by the input x, while $\mathbf{a}_v(x')$ denotes those from a corrupted prompt x'. The corrupted input x' is specifically constructed to remove key task-related information, thereby allowing us to isolate the causal necessity of each edge. The term $\nabla_{\mathbf{a}_v}\mathcal{L}(x)$ represents the sensitivity gradient of the metric with respect to the input of the node v on clean data; $\langle\cdot,\cdot\rangle$ denotes the inner product; $\mathcal{D}$ represents the task-specific dataset; T is the sequence length; i denotes the token position. Similar to Syed et al. (2024), we implement the difference in logit values corresponding to ground truth and the predicted output by the LLMs, as the primary metric $\mathcal{L}$ to measure the model's confidence. For specific tasks, this metric is defined as follows. Since EAP relies on a first-order approximation rather than an actual intervention, we validate the identified components with activation patching. Specifically, we replace their clean-run contributions to the child inputs with the corresponding contributions from the corrupted run. Across all models and tasks, this intervention degrades task performance substantially more than intervening on the same number of randomly selected edges. Further details are in Appendix D.2.

Metric for Binary Sentiment Classification ($\mathcal{L}_{\text{Sent}}$) Here we measure the difference between the logit of the target label y^* (denoted as z_{y^*}) and that of the highest-scoring incorrect label for a particular input x. If certain edges are important, then corrupting them should not only alter the model's output significantly but also reduce this logit difference. Mathematically $\mathcal{L}_{\text{Sent}}$ is defined in Equation 5.

$$\mathcal{L}_{\text{Sent}} = z_{y^*} - \max_{j\neq y^*} z_j \quad (5)$$

where j ranges over all class labels.

Metric for Generative Tasks ($\mathcal{L}_{\text{QA, MT}}$) For generative tasks, token sequences are produced under a teacher-forcing regime. At each decoding step, we compute the logit difference between the ground-truth token (GT_i) and the highest-scoring incorrect token within the top-K predicted candidates. This logit difference is then aggregated across all n decoding steps, where n denotes the total number of tokens in the ground-truth sequence (GT). Equation 6 describes the mathematical formulation for this metric.

$$\mathcal{L}_{\text{QA, MT}} = \sum_{i=1}^{|GT|}\left(z^{(i)}_{GT_i} - \max_{j\in K_i\setminus\{GT_i\}} z^{(i)}_j\right) \quad (6)$$

where j ranges over the top-K candidate tokens at that step excluding the ground-truth token.

Across all generative task datasets, the output sequence lengths exhibit low variability, with an average standard deviation of 2.12. This indicates that token length variation is minimal and unlikely to introduce any bias in the evaluation metric $\mathcal{L}_{\text{QA, MT}}$.

4 Experiment Setup

Dataset Description. Existing research (Alves et al., 2024; Subramonian et al., 2023) showed that the three tasks mentioned above are among the most widely used NLP tasks. Hence, we focused particularly on these three tasks in our research scope. For **Sentiment Classification** task, we used two datasets: Large Yelp Review (Yelp) Dataset (Zhang et al., 2015) and Stanford Sentiment Treebank (SST-2) (Socher et al., 2013). For **Question Answering** task, we used two benchmark reading-comprehension-style datasets: the Stanford Question Answering Dataset (SQuAD) v1.1 (Rajpurkar et al., 2016) and Conversational Question Answering (CoQA) (Reddy et al., 2019). For **Machine Translation** task, we used KDE4 dataset

(Tiedemann, 2012) and Tatoeba dataset (Artetxe and Schwenk, 2019). The details of the datasets are given in Appendix B.

Preprocessing. As described in § 3.2, the EAP algorithm used to estimate the EAP importance of different components requires corrupted inputs corresponding to each sample input. To generate the corrupted sentences for the sentiment classification task, we masked the tokens most indicative of sentiment in each sentence using the '<mask>' token. These sentiment-bearing tokens were automatically identified using the Meta-Llama-3-8B-Instruct model. Similarly, for the question answering task, the answer span within the context passage was replaced with the '<mask>' token. For the machine translation task, where most source tokens contribute significantly to the output, we replaced all tokens in the English input with '<mask>' to simulate maximal information loss. Examples of generated corrupted data are shown in Appendix C. Although the corrupted token strategies differ across tasks, they follow the same principle of removing information directly relevant to the task output. We do not assume that the resulting corruption severity is identical across tasks; instead, EAP is applied within each task to rank task-relevant components, and our cross-task analyses compare these identified component sets. We also test sensitivity to the corruption token in Appendix D.3. Replacing '<mask>' with several alternative tokens produces strongly correlated edge rankings ($\rho = 0.71$–0.72), suggesting that the identified components are not specific to the '<mask>' token.

For EAP algorithm described in § 3.2, we considered only the components within top 400 edges for our research scope since we found that most of the important components across all of the LLMs and for all the tasks lie within this range. Figure 7 in Appendix D.1 shows the contribution of these components across different tasks. It can be observed from Figure 7 that using only components from top-400 edges preserves at least 90% of the original task performance across the evaluated settings.

FT Setup. We used GPT-2 Small, Llama-3.2-1B, Qwen2-0.5B and Llama-2-7B model in our experiment. All four models were fine-tuned in a full FT setup, i.e., all model parameters were updated. In Appendix F, we described in detail other parameters like batch size, epoch used in FT.

5 Results

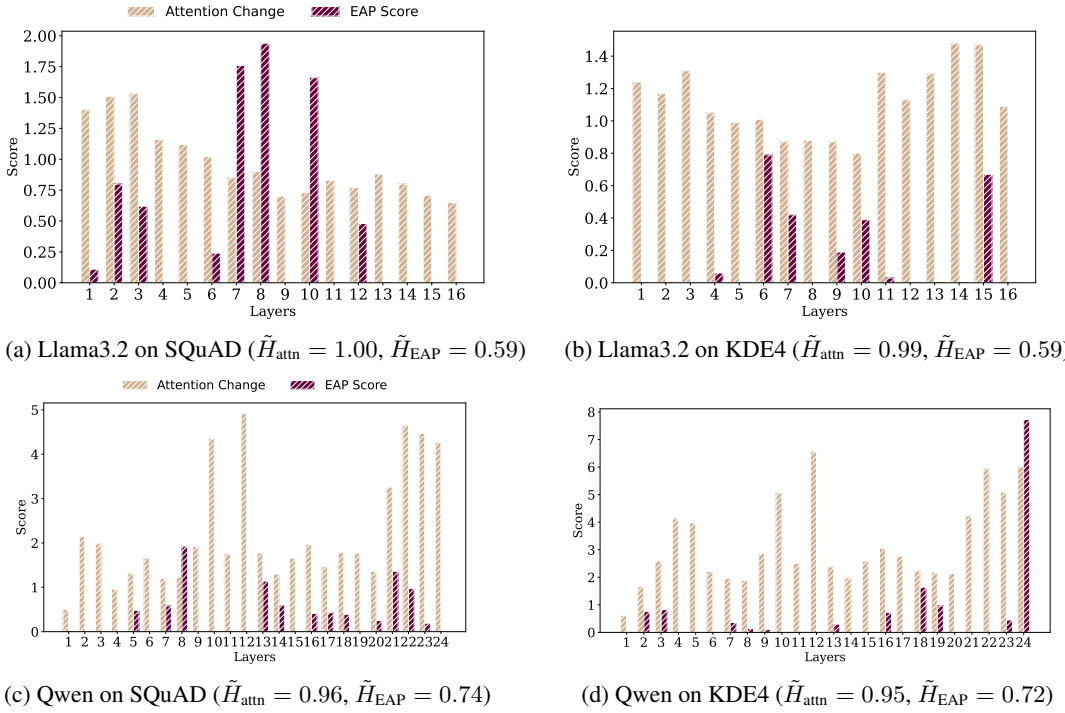
We present results along three dimensions, namely, (i) the distribution of changes induced by FT, (ii) the distribution of causally relevant components after FT, and (iii) the relationship between overlap of EAP-identified components and the cross-task performance transfer capability of the fine-tuned model.

5.1 Heterogeneity of Fine-tuning Dynamics

To demonstrate the non-uniformity of FT dynamics, Figure 2 illustrates the changes in attention patterns across layers (computed using Equation 1), while Figure 3 shows the variation in the amount of information encoded in each layer after FT.

Two key observations emerge from Figure 2. First, attention pattern change is specific to a particular task and model, and it is not uniform. We quantify this non-uniformity using the normalised layer-wise entropy $\tilde{H} \in [0, 1]$ of each distribution (defined in Appendix E). The different values of attention entropy across different variations (i.e., different models and tasks) also further confirm this observation. Secondly, we observe that smaller models exhibit a substantially greater magnitude of representational change during FT compared to larger models. For instance, the average KL-divergence values in Llama-3.2-1B span a higher range (approximately 7–10), whereas for Llama-2-7B they remain significantly lower (approximately 0–1). This observation is broadly consistent with prior work on representation preservation and parameter-efficient adaptation (Aghajanyan et al., 2021), which suggests that larger pretrained models possess richer reusable features and therefore require comparatively smaller representational modifications during downstream FT. For the rest of the models and the datasets, refer to Figure 12 and Figure 13 in Appendix L.

Figure 3 shows how task-related information encoded in different layers change after FT. The main observations from it are as follows. Firstly, both for pretrained and finetuned setup for all the tasks and for all the models, the probing performance of the initial layers is worse than later layers. Consequently, it can be said that the initial layer representations do not contain much information about the task. This observation is similar to Zhao et al. (2024) who show that in general, initial layers of LLMs store core language capabilities and the later layer has more derived properties. Sec-



(a) Llama3.2 on SQuAD ($\tilde{H}_{\text{attn}} = 1.00$, $\tilde{H}_{\text{EAP}} = 0.59$)

(b) Llama3.2 on KDE4 ($\tilde{H}_{\text{attn}} = 0.99$, $\tilde{H}_{\text{EAP}} = 0.59$)

(c) Qwen on SQuAD ($\tilde{H}_{\text{attn}} = 0.96$, $\tilde{H}_{\text{EAP}} = 0.74$)

(d) Qwen on KDE4 ($\tilde{H}_{\text{attn}} = 0.95$, $\tilde{H}_{\text{EAP}} = 0.72$)

Figure 2: Layer-wise distribution of average KL divergence in the attention matrices before and after FT, and summed EAP scores across Fine-tuned Qwen2 and llama-3.2-1B. For GPT-2 Small and Llama-2-7B refer to Figure 12 in Appendix L; results for Qwen2 and Llama-3.2-1B on the selected remaining datasets are shown in Figure 13 in Appendix L. Values in parentheses are the normalised layer-wise entropy ($\tilde{H}_{\text{attn}}$, $\tilde{H}_{\text{EAP}}$); the consistently lower $\tilde{H}_{\text{EAP}}$ indicates that EAP importance is more localised across layers. See Appendix E for the entropy definition.

ondly, the probing performance in the FT setup only changes at later layers mostly. This observation concludes that FT doesn't affect the core capabilities of the model. Thirdly, it can be observed clearly that for all the models and for all the tasks the probing performance has increased after FT. It indicates that FT increases the amount of task-level information encoded in each layer. For GPT-2 Small model and the rest of the datasets, refer to Figure 14 in Appendix L. Since probing results are output-mediated and capture layer-wise decodability, we further investigate a direct measure of representational geometry change in each layer rather than passing through the output head of model in Appendix K. It shows that across all models and tasks, the most substantial layer-wise representational changes occur in the later layers (in the order of 0.7 to 0.9), specifically from approximately the 90th percentile layer to the final layer.

Overall, from both Figure 2, Figure 3 and Appendix K, it is evident that the internal change in the model is not uniform across all the layers and it is valid across all the models and tasks.

5.2 Localisation of EAP-Identified Components

Figure 2 presents the layer-wise distribution of the total importance of causally relevant components, where importance scores are computed using EAP equation described in Equation 3. Equation 3 provides score to each edge and then for each layer we show the sum of the scores corresponding to the top-400 edges belonging to that layer.

Two key observations emerge from the EAP score distribution in Figure 2. First, a substantial number of layers exhibit negligible or no contribution to the set of causally relevant components. The entropy score of the EAP score distribution further supports this observation. In contrast to the attention change, the EAP-score distribution has a lower entropy ($\tilde{H}_{\text{EAP}} = 0.40$–$0.74$; see Appendix E). This pattern is consistent across all models and tasks, suggesting that EAP-identified components are highly localised. Second, the layers that undergo the largest changes in attention patterns do not necessarily correspond to those with the highest EAP importance. This indicates that the magnitude of internal change is only weakly

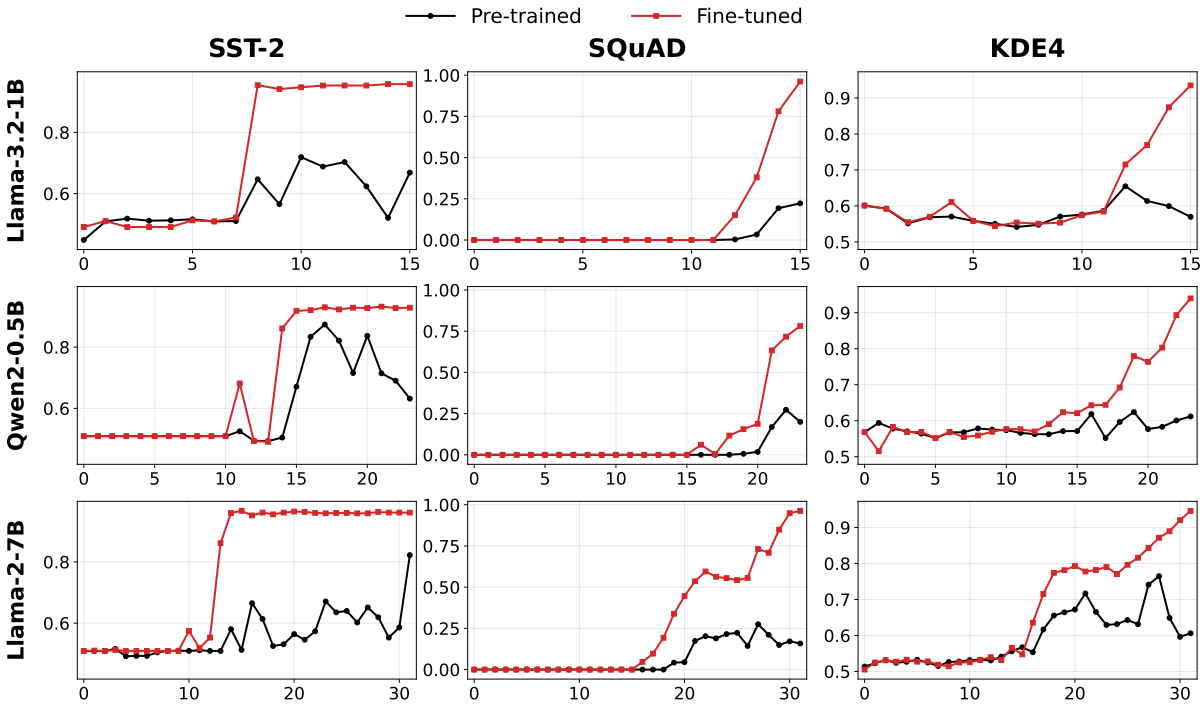
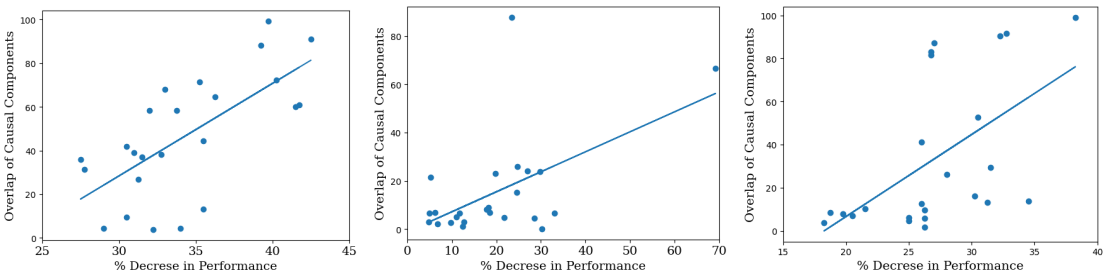


Figure 3: Probing results of pretrained model vs. fine-tuned model across all the layers of Llama-3.2-1B, Qwen2-0.5B, Llama-2-7B for Sentiment Classification (SST-2), Question Answering (SQuAD), and Machine Translation (KDE4). Probing performance for each task is measured by the corresponding task metric as described in Section 3.1. Higher values indicate that the correct task output is more readily recoverable from that layer's representation via the logit lens. The complete layer-wise probing results for all four models across all six datasets are shown in Figure 14 in Appendix L. (X-axis: layer index, Y-axis: binary accuracy for SST-2, SQuAD F1 Score for SQuAD, BERTScore for KDE4).



(a) Qwen2, $\rho = 0.7, p = 0.0002$ (b) Llama-3.2-1B, $\rho = 0.6, p = 0.005$ (c) Llama-2-7B $\rho = 0.6, p = 0.005$

Figure 4: Correlation between Cross-task Performance and Important Component Overlap for Llama-3.2-1B, Qwen2-0.5B and Llama-2-7B. ρ is correlation coefficient and p-values show the significance of the correlation coefficient. The result for GPT-2 Small is provided in Figure 10, Appendix L.

correlated with causal relevance.

5.3 Cross-task Performance Transfer

Here, we analyse whether FT a model on Task 1 leads to improvements or degradation in performance on Task 2, where the two tasks are substantially different in nature (e.g., sentiment analysis and machine translation). More specifically, our objective is to examine the relationship between overlap of EAP-identified components and cross-task performance transfer. In particular, we investigate whether the overlap in EAP-identified components between Task 1 and Task 2 influences performance on Task 2 when the model is fine-tuned on Task 1. To this end, Figure 4 shows the correlation between a decrease in performance and overlap of top 400 EAP-identified components, and for each task, we consider four distinct tasks to compute both performance changes and overlap of EAP-identified components. In total, this results in 24 task pairs

for comparison. The list of tasks is in the Appendix, Table 6 "Dissimilar Task Analysis".

Two main observations emerge from Figure 4. Firstly, we observe that FT a model on Task 1 generally leads to a degradation in performance on Task 2, relative to the pretrained model's performance on Task 2. Secondly, we observe a positive correlation across all models (Qwen2, Llama-3.2-1B, and Llama-2-7B) between the degree of overlap of EAP-identified components and the extent of performance degradation. See result for GPT-2 Small in Figure 10, Appendix L. This suggests that higher overlap between the EAP-identified components of Task 1 and Task 2 is associated with greater performance decline on Task 2 when the model is fine-tuned on Task 1 and vice versa. A plausible explanation for this phenomenon is that FT adapts a specific set of components to the requirements of Task 1, which is unlikely to generalise well to Task 2 when the two tasks are different in nature. To validate that the shared components are responsible for the degradation directly rather than correlational, we perform an intervention in Appendix J: we identify the components that are shared between the two tasks, freeze them, and fine-tune only the remaining parameters on Task 1. Protecting the overlapping components during fine-tuning recovers part of the lost performance on Task 2. So protecting them provides intervention-based evidence that the overlapping components contribute to the observed interference.

6 Causal Structure Analysis

To further investigate the important EAP-derived components, we analyse the distribution of different architectural elements, specifically embedding nodes, attention heads, MLP neurones, logit nodes, and among the EAP-derived components selected for each task. Figure 5 presents pie charts illustrating the composition across three tasks in Llama-3.2-1B. More analysis on the rest of the models and datasets are shown in Figure 8 in Appendix G. The results show a consistent pattern across tasks and models: the components spanned by the top-400 EAP-ranked edges are predominantly composed of MLP neurones and attention heads, with relatively limited contribution from other component types. To avoid the frequency bias from the component distribution of the original model, we also provided normalised frequency of different components in Appendix G. It can be seen that the presence of MLPs and logits are multiplied in the normalised frequency.

Given the prominence of attention heads within these important components, we further investigated the specific involvement of *induction heads* (Olsson et al., 2022), components responsible for pattern matching within input context. We investigated induction heads following the synthetic repeated-sequence protocol of Olsson et al. (2022) (implemented via TransformerLens, Nanda and Bloom, 2022). Table 1 shows the Jaccard overlap between the detected induction heads and the task-specific causally important attention heads, computed as the size of their intersection divided by the size of their union. The detection details and the full metric table are provided in Appendix I.

We observe three key trends from Table 1. Firstly, three of the four models exhibit their highest overlap on question answering, with CoQA yielding the maximum overlap for GPT-2 Small (16.66%), Llama-3.2-1B (6.16%), and Llama-2-7B (4.34%). Qwen2-0.5B is the exception, reaching its highest overlap on KDE4 (7.19%). Secondly, the results suggest a task-dependent pattern in the reuse of induction heads. Question answering tends to show relatively higher overlap with induction heads for GPT-2 Small, Llama-3.2-1B and Llama-2-7B. This may be related to the reading-comprehension nature of the QA datasets used in our study, where answers must be identified from or grounded in the input context. Such behaviour is conceptually similar to the pattern-matching mechanisms associated with induction heads (Olsson et al., 2022). Thirdly, this tendency is not consistent across all models and datasets. Qwen2-0.5B exhibits its highest overlap on KDE4, while its overlap on Tatoeba remains considerably lower. Likewise, MT shows relatively low overlap for the two Llama models. These results suggest that induction-head involvement is not determined solely by task family, but may also depend on model- and dataset-specific factors.

7 Conclusion

We studied how FT reshapes LLM internals along two aspects: the relationship between representational change and the EAP-identified components driving task performance, and the effect of shared EAP-identified components on cross-task transfer. Across four models and six datasets, we find that the layers undergoing the largest representational and attention changes do not necessarily coincide

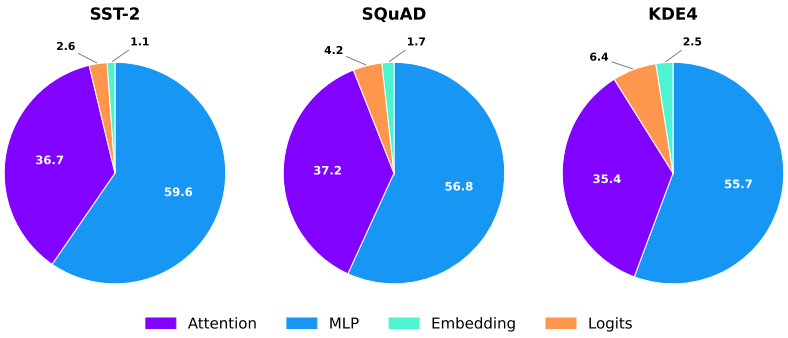


Figure 5: Distribution percentage of important components among the top-400 edges in Llama-3.2-1B across three tasks: Sentiment Classification (SST-2), Question Answering (SQuAD). Machine Translation (KDE4). The full results for four models and six datasets are given in Figure 8 in Appendix G.

Model	Task	#Induction Heads	#Attention Heads	Overlap(%)
GPT-2 Small	Sentiment (Yelp)	26	101	8.54%
	Sentiment (SST-2)	24	80	1.96%
	QA (SQuAD)	24	98	12.96%
	QA (CoQA)	24	109	**16.66%**
	MT (KDE4)	24	99	7.89%
	MT (Tatoeba)	24	93	7.33%
Llama-3.2-1B	Sent (Yelp)	62	151	3.39%
	Sent (SST-2)	58	146	1.49%
	QA (SQuAD)	62	158	5.76%
	QA (CoQA)	62	162	**6.16%**
	MT (KDE4)	64	167	2.21%
	MT (Tatoeba)	57	132	2.71%
Qwen2-0.5B	Sent (Yelp)	40	112	2.70%
	Sent (SST-2)	35	83	1.85%
	QA (SQuAD)	36	97	3.10%
	QA (CoQA)	39	110	3.75%
	MT (KDE4)	33	116	**7.19%**
	MT (Tatoeba)	40	75	1.76%
Llama-2-7B	Sent (Yelp)	53	263	1.28%
	Sent (SST-2)	52	94	0.00%
	QA (SQuAD)	15	270	3.70%
	QA (CoQA)	61	227	**4.34%**
	MT (KDE4)	58	166	0.00%
	MT (Tatoeba)	49	88	0.00%

Table 1: Overlap of Induction head vs. Attention heads derived from the top-400 EAP-ranked edges across all models for Sentiment Analysis (Sent), Question Answering (QA) and Machine Translation (MT). For each model the highest overlap is boldfaced and the highest overlap percentage across all the models is underlined.

with those carrying the highest EAP importance. More generally, the magnitude of internal change is only weakly correlated with causal relevance. On cross-task transfer, we find that a high overlap in EAP-identified components between two tasks does not aid generalisation; instead, it is associated with greater performance degradation on the second task, which suggests that components adapted to one task do not transfer benignly to a substantially different one. Together, these results indicate that effective fine-tuning is governed by a small, localised set of EAP-identified components rather than by broad representational shifts.

In the future, we plan to extend this work to a larger number of tasks. Our findings related to different aspects of localisation and generalisation can also be used to design efficient and effective strategies of FT in NLP.

Limitations

We restrict our analysis to three NLP task families (sentiment classification, question answering, and machine translation). Although these tasks encompass diverse input–output structures, they do not represent the full spectrum of real-world applications, such as reasoning-intensive or multimodal tasks, which limits the transferability of our findings. We also examined only GPT-2 Small and Llama-3.2-1B, Llama-2-7B and Qwen2-0.5B due to the shortage of computing resources, limiting generalisability to larger models that may exhibit different FT behaviours. Further work is needed to investigate more models.

Ethical Considerations

This work involves non-sensitive publicly accessible datasets. This work on mechanistic interpretability presents a dual-use dilemma: while understanding FT internals can improve model safety and efficiency, it could also enable adversarial exploitation. To mitigate this risk, we advocate for responsible disclosure practices and recommend that future work on efficient FT methods include built-in safety mechanisms.

Acknowledgments

This research was supported by the University of Liverpool and the Engineering and Physical Sciences Research Council (EPSRC). We thank the School of Computer Science and Informatics at the University of Liverpool for providing access

to computational resources. The authors also acknowledge the anonymous reviewers for their constructive suggestions.

Danushka Bollegala holds concurrent appointments as a Professor at University of Liverpool and as an Amazon Scholar. This paper describes work performed at the University of Liverpool and is not associated with Amazon.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Diego Alves, Marko Tadić, and Georg Rehm. 2024. Which domains, tasks and languages are in the focus of NLP research on the languages of Europe? In *Proceedings of the Second International Workshop Towards Digital Language Equality (TDLE): Focusing on Sustainability @ LREC-COLING 2024*, pages 18–32.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, Online. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286.

Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. 2024. Saullm-7b: A pioneering large language model for law.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*.

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.

Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert Dick, Hidenori Tanaka, Tim Rocktaeschel, Edward Grefenstette, and David Krueger. 2024. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. In *International Conference on Learning Representations*, volume 2024, pages 9944–10015.

Che Jiang, Biqing Qi, Xiangyu Hong, Dayuan Fu, Yang Cheng, Fandong Meng, Mo Yu, Bowen Zhou, and Jie Zhou. 2024. On large language models' hallucination with regard to known facts. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1041–1053, Mexico City, Mexico. Association for Computational Linguistics.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang.

2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Yongqi Leng and Deyi Xiong. 2025. Towards understanding multi-task learning (generalization) of llms via detecting and exploring task-specific neurons. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2969–2987.

Yuyan Liu, Sirui Ding, Sheng Zhou, Wenqi Fan, and Qiaoyu Tan. 2024. Moleculargpt: Open large language model (llm) for few-shot molecular property prediction.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6).

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. 2025. Reinforcement learning finetunes small subnetworks in large language models. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 132119–132138. Curran Associates, Inc.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`.

nostalgebraist. 2020. Interpreting gpt: The logit lens. `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`. LessWrong blog post.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *arXiv (Cornell University)*.

Haowen Pan, Yixin Cao, Xiaozhi Wang, Xun Yang, and Meng Wang. 2024. Finding and editing multi-modal neurons in pre-trained transformers. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1012–1037, Bangkok, Thailand. Association for Computational Linguistics.

Abhishek Panigrahi, Nikunj Saunshi, Haoyu Zhao, and Sanjeev Arora. 2023. Task-specific skill localization in fine-tuned language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27011–27033. PMLR.

Nikhil Prakash, Tamar Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. 2024. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *International Conference on Learning Representations*, volume 2024, pages 8057–8082.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Yi Ren and Danica J. Sutherland. 2025. Learning dynamics of LLM finetuning. In *The Thirteenth International Conference on Learning Representations*.

William Rudman, Catherine Chen, and Carsten Eickhoff. 2023. Outlier dimensions encode task specific knowledge. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14596–14605, Singapore. Association for Computational Linguistics.

Dong Shu, Haoran Zhao, Xukun Liu, David Demeter, Mengnan Du, and Yongfeng Zhang. 2024. Lawllm: Law large language model for the us legal system. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 4882–4889. ACM.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Arjun Subramonian, Xingdi Yuan, Hal Daumé III, and Su Lin Blodgett. 2023. It takes two to tango: Navigating conceptualizations of NLP tasks and measurements of performance. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3234–3279.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416.

Amit Tewari. 2024. Legalpro-bert: Classification of legal provisions by fine-tuning bert large language model.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2214–2218, Istanbul,

Turkey. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding skill neurons in pre-trained transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11132–11152, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xu Wang, Yan Hu, Wenyu Du, Reynold Cheng, Benyou Wang, and Difan Zou. 2025. Towards understanding fine-tuning mechanisms of LLMs via circuit analysis. In *Forty-second International Conference on Machine Learning*.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Weidi Xie, and Yanfeng Wang. 2024. Pmc-llama: toward building open-source language models for medicine. *Journal of the American Medical Informatics Association*, 31(9):1833–1843.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah Goodman. 2023. Interpretability at scale: Identifying causal mechanisms in alpaca. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Haoyun Xu, Runzhe Zhan, Yingpeng Ma, Derek F Wong, and Lidia S Chao. 2025a. Let's focus on neuron: Neuron-level supervised fine-tuning for large language model. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9393–9406.

Yang Xu, Yi Wang, and Hao Wang. 2025b. Tracking the feature dynamics in llm training: A mechanistic study.

Zeping Yu and Sophia Ananiadou. 2025. Locate-then-merge: Neuron-level parameter fusion for mitigating catastrophic forgetting in multimodal LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7065–7078, Suzhou, China. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Zheng Zhao, Yftah Ziser, and Shay B Cohen. 2024. Layer by layer: Uncovering where multi-task learning happens in instruction-tuned large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15195–15214.

A Description of Notations

In Figure 1, each panel shows the top-12 highest-$|s_e|$ edges of the EAP causal-attribution graph for one task. The visual encoding is the same across all panels.

Nodes are the transformer components touched by those edges, colour-coded by category: **input** (embedding), **mℓ** (MLP of layer ℓ), **aℓ.hi** (attention head i of layer ℓ; Q/K/V ports collapsed), and **logits** (unembedding).

Edges are coloured by the sign of their attribution score s_e: red for $s_e > 0$ (the edge helps the task loss) and blue for $s_e < 0$ (the edge hurts it). Line width is proportional to $|s_e|$, so thicker edges are more causally important to the task; the thickness scale is shared across all panels, so importance is also comparable across tasks.

B Dataset Description

Sentiment Classification Dataset. In the sentiment classification task, we implemented two datasets: Large Yelp Review (Yelp) Dataset (Zhang et al., 2015) and Stanford Sentiment Treebank (SST-2) (Socher et al., 2013). The Yelp dataset is used for binary sentiment classification; it contains $560,000$ training data and $38,000$ test data of highly polar Yelp reviews. The dataset is extracted from the Yelp Dataset Challenge by 2015 data. SST-2 corpus, which we use the version integrated into the GLUE benchmark (Wang et al., 2018), consists of movie reviews originally extracted from the

Task Information	Original Input Sequence	Corrupted Input Sequence
Task 1: Sentiment Classification (SST-2/Yelp)		
Info Column: *Sentiment Label;* ***Input Column:*** *Review Text*		
Label: Negative	a sometimes **tedious** film.	a sometimes **X** film.
Label: Positive	among the year's most **intriguing** explorations of alienation.	among the year's most **X** explorations of alienation.
Task 2: Machine Translation (Tatoeba/KDE4)		
Info Column: *Target Translation (French);* ***Input Column:*** *Source English*		
Target: Il est resté silencieux.	He remained dumb.	x x xx
Task 3: Question Answering (CoQA/SQuAD)		
Info Column: *Question;* ***Input Column:*** *Context Snippet (Answer Span Bolded)*		
Q: What color was Cotton?	... lived a little **white** kitten named Cotton. Cotton lived high up...	... lived a little **X** kitten named Cotton. Cotton lived high up...

Table 2: Examples of corrupted samples across three downstream tasks.

Rotten Tomatoes website. It provides a binary classification benchmark consisting of $67,349$ training, 872 validation and $1,821$ test samples. In both datasets, each sentence is labelled as either positive or negative. For Yelp, we first select $11,000$ examples from the training split and apply a 90/10 split, resulting in $9,900$ examples used for fine-tuning. Evaluation is performed on $1,000$ examples selected from the test split. For SST-2, the GLUE test labels are not public, so we evaluate on the full labeled validation split. Training uses the first $15,000$ examples of the SST-2 train split.

Question Answering Dataset. In the question answering task, we used two benchmark reading-comprehension-style question answering datasets: the Stanford Question Answering Dataset (SQuAD) v1.1 (Rajpurkar et al., 2016) and CoQA (Reddy et al., 2019). Both datasets provide a context passage as part of the input, and the answer is expected to be grounded in the given context rather than generated solely from the model's parametric knowledge. In the train/validation splits used in our experiments, SQuAD v1.1 contains 87,599 training and $10,570$ validation question-answer pairs, totaling $98,169$ QAs. The dataset is based on Wikipedia articles, where each question is answered using a text span from the corresponding context passage. CoQA contains $108,647$ training and $7,983$ validation QAs, totaling $116,630$ QAs. It is designed for conversational question answering across multiple domains and emphasises multi-turn interactions with context-dependent responses. For SQuAD, we select $11,000$ examples from the training split and apply a 90/10 split, resulting in 9,900 examples used for fine-tuning. We evaluate on $1,000$ examples from the SQuAD validation split. For CoQA, we flatten the training examples, select $40,000$ instances, and apply a 90/10 split, resulting in $36,000$ examples used for fine-tuning. We evaluate on $1,000$ flattened examples from the CoQA validation split.

Machine Translation Dataset. We employed two parallel corpora for machine translation model: the KDE4 dataset (Tiedemann, 2012) and Tatoeba dataset (Artetxe and Schwenk, 2019). The KDE4 dataset consists of a bilingual corpus derived from the localisation files of the KDE Software Environment, with English and French pairs of sentences. It has almost $200,000$ sentence pairs. The English sentences in the KDE4 dataset have an average length of approximately 12.4 words, while the French translations average 14.7 words, reflecting the slightly more verbose nature of French in technical contexts. The Tatoeba dataset is a community-driven, open-source collection of sentence translations across multiple languages, including English and French. It is widely used to evaluate machine translation models because of its diversity and naturalistic content. This dataset contains approximately $300,000$ English-French sentence pairs. For KDE4, we select 30,000 examples and apply a 90/10 split, resulting in $27,000$ examples used for fine-tuning. Evaluation is conducted on a disjoint slice of $1,000$ examples from the KDE4 training corpus. For Tatoeba, we select $40,000$ examples and apply a 90/10 split, resulting in $36,000$ examples used for fine-tuning. Evaluation is conducted on a disjoint slice of $1,000$ examples from the Tatoeba training corpus.

C Corrupted Dataset Generation Details

To analyse the localisation of task-specific knowledge, we constructed corrupted versions of the datasets by masking tokens that are critical for task performance. This section details the corruption strategy and provides qualitative examples for each task.

Sentiment Word Identification Details Instead of using a fixed lexicon, we leveraged `Meta-Llama-3-8B-Instruct`[2] to dynamically identify sentiment-bearing tokens relevant to the specific context. The process involved three steps:

Sample Selection: We first filtered for short sentences (5–10 tokens) from the validation set where our fine-tuned model originally made correct predictions. This ensures that the identified words are indeed responsible for the model's correct classification behaviour.

LLM Querying: We constructed a few-shot prompt containing three examples (positive, negative, and neutral) to guide Llama-3 in extracting the most indicative sentiment words. The exact prompt template used is shown in Figure 6.

Response Parsing: The generated output was parsed to extract a comma-separated list of words. We strictly validated that the extracted words appeared in the original text and filtered out function words or tokens with fewer than 3 characters. Up to three unique words were selected per sample.

Table 2 presents a comprehensive side-by-side comparison of original versus corrupted inputs across the three downstream tasks.

Prompt Used for Sentiment Word Identification

System Instruction:
You are a helpful assistant that extracts sentiment words from reviews.

User Input:
Given that this review has a `{sentiment}` sentiment, identify up to three words that most strongly indicate this `{sentiment}` sentiment. Only select words that appear in the review text. Return the words in a comma-separated list. If no words strongly indicate the sentiment, return an empty list.
Examples:

- **Review:** The food was amazing and the service was excellent!
  Sentiment: positive → **Output:** amazing, excellent
- **Review:** Terrible experience, never coming back.
  Sentiment: negative → **Output:** terrible
- **Review:** It was okay, nothing special.
  Sentiment: negative → **Output:** *(empty)*

Current Task:
Review: `{input_text}`
Sentiment: `{label}`
Output:

Figure 6: The few-shot prompt template used to extract sentiment-sensitive words via Llama-3-8B-Instruct. We dynamically insert the specific sentiment label (positive/negative) and the review text into the placeholders (marked in blue) for each query.

D Validation of EAP-identified components

This section provides three additional analyses to demonstrate the faithfulness and robustness of the EAP edge rankings: a) performance comparison of retaining the top-k EAP-ranked edges, retaining the same number of randomly selected edges and retaining all the edges at different k values, which is to prove the rationale for setting the top-400 edges in our research; b) performance comparison that patching the activations of the top-400 EAP-ranked edges against that caused by patching on these same number of randomly selected components, which is to strengthen the core causal claim; c) compare the Spearman rank correlation of resulting edge-important scores between corruption token "<mask>" and other tokens, which is to evaluate the robustness of corruption token selection.

D.1 Sufficiency and the Choice of k

Figure 7 shows the effect of using only top-k edges (k = 100, 200, 300, 400, 500, 600, 700) on the performance of GPT-2 small and Llama-3.2-1B, Qwen2-0.5B and Llama-2-7B in Yelp dataset based sentiment classification task. More specifically, we retain only the top-k edges identified for each task and set the contributions of all remaining edges to zero. It can be observed that using only 400 edges the 90% of the model's original performance can be attained. This shows the faithfulness of the top-400 edges identified by EAP. We can also observe that the identified top-k edges have always performed better than the randomly selected k edges.

[2] `https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`

Model	Task	EAP	Random	Δ
GPT-2 Small	Sent (Yelp)	85%	61%	24
	QA (SQuAD)	80%	55%	25
	MT (KDE4)	80%	67%	13
Llama-2-7B	Sent (Yelp)	81%	58%	23
	QA (SQuAD)	84%	52%	32
	MT (KDE4)	79%	54%	25
Qwen2-0.5B	Sent (Yelp)	83%	59%	24
	QA (SQuAD)	82%	55%	27
	MT (KDE4)	84%	56%	28
Llama-3.2-1B	Sent (Yelp)	81%	61%	20
	QA (SQuAD)	86%	52%	34
	MT (KDE4)	82%	44%	38

Table 3: Performance drop caused by intervening on the top-400 EAP-ranked edges versus the same number of randomly selected edges. Random results are averaged over 20 seeds. Δ is the absolute difference in percentage points. The difference is significant for every model–task pair (two-sided Wilcoxon signed-rank test, $p < 0.05$).

Corruption token	Spearman ρ vs. <mask>
neutral	0.71
blank	0.72
abc	0.71

Table 4: Spearman rank correlation between edge-importance scores obtained with alternative corruption tokens and those obtained with <mask>, computed over all edges of GPT-2 Small on the sentiment classification task.

D.2 Necessity of Corrupting the Top-400 Edges

To verify that the observed degradation is specific to the top-400 EAP-ranked edges, rather than a consequence of intervening on a large number of edges, in Table 3, we compare the performance drop caused by intervening on these edges against that caused by intervening on the same number of randomly selected edges. In both cases, we intervene by activation patching. For each selected edge, its clean-run contribution to the child input is replaced by the corresponding contribution from the corrupted run. The random baseline is constructed by uniformly sampling 400 edges from the full set of model edges, and the results are averaged over 20 random seeds.

D.3 Robustness to the Corruption Token

In Table 4, we run EAP on GPT-2 fine-tuned on the sentiment classification task while varying the corruption token (i.e., "neutral", "blank", "abc"), and measure the Spearman rank correlation between the resulting edge-importance scores over the full model. The edge rankings are highly correlated across tokens. It can be seen that for all the corrupted token variant it has a strong correlation with the edges obtained using "<mask>" token.

E Entropy of the Attention-Change and EAP Distributions

We summarise how localised each distribution in Figure 2 is by a single entropy value, computed as follows. For a model with L layers, each distribution provides one value per layer: $x_1, \ldots, x_L$, where x_ℓ is either the average attention change ΔA_ℓ (Equation 1) or the summed EAP score at layer ℓ. We treat these per-layer values as a distribution over layers by normalising them,

$$p_\ell = \frac{x_\ell}{\sum_{j=1}^{L} x_j}, \qquad \sum_{\ell=1}^{L} p_\ell = 1,$$

and then collapse the L per-layer values into a single scalar via the Shannon entropy (in bits) and its normalised form,

$$H = -\sum_{\ell=1}^{L} p_\ell \log_2 p_\ell, \qquad \tilde{H} = \frac{H}{\log_2 L},$$

with the convention $0 \log_2 0 = 0$. The summation over layers is what reduces each per-layer distribution to one number, so every distribution yields a single entropy value. Normalising by $\log_2 L$ (the maximum entropy, attained by a uniform distribution) maps $\tilde{H}$ to $[0, 1]$ and makes it comparable across models of different depths: $\tilde{H} \to 1$ means the change is spread evenly across all layers, while $\tilde{H} \to 0$ means it is concentrated in a few layers. Thus, each subplot is summarised by the pair $(\tilde{H}_{\text{attn}}, \tilde{H}_{\text{EAP}})$ reported in the caption of Figure 2. Across all models and tasks, $\tilde{H}_{\text{EAP}}$ is substantially lower than $\tilde{H}_{\text{attn}}$, confirming that EAP-identified components are concentrated in a small number of layers.

F Hyperparameter Setup in FT

To verify our findings across various settings, we conduct experiments on four language model families: **GPT-2 Small (124M)**, **Qwen-2-0.5B**, **Llama-3.2-1B** and **Llama-2-7B**. All experiments were run on a single NVIDIA A100 80GB GPU. The total computational budget was approximately 14 GPU hours. The detailed fine-tuning parameters are in Table 5.

Parameter	Llama-2 (7B)	Llama-3.2 (1B)	Qwen2 (0.5B)	GPT-2 (Small)
Core Optimization				
Optimizer	AdamW (fused)	AdamW (fused)	AdamW (fused)	AdamW (fused)
Learning Rate	2×10^{-5}	2×10^{-5}	2×10^{-5}	$5 \times 10^{-5\dagger}$
Weight Decay	0.0	0.0	0.0	0.0
Warmup Ratio	0.03	0.03	0.03	0.03
LR Scheduler	Cosine	Cosine	Cosine	Cosine
Precision	bf16	bf16	bf16	bf16
Effective Batch Size				
SST-2 / Yelp	16 / 8	32	32 / 16	32
QA (SQuAD/CoQA)	8	32	32 / 16	32
MT (KDE4/Tatoeba)	8	32	32	32
Training Epochs				
SST-2 / Yelp	2	3	3	3
QA (SQuAD/CoQA)	2 / 1	2	2	3
MT (KDE4/Tatoeba)	1	3	3	3
Sequence Length				
SST-2 / Yelp	256	512	512	512
QA (SQuAD/CoQA)	1024	1024	1024	1024
MT (KDE4/Tatoeba)	256	256	256	256

Table 5: Complete hyperparameter settings across models and tasks. All four models are fully fine-tuned. Effective batch size is per-device batch size $\times$ gradient-accumulation steps. †GPT-2 SQuAD/CoQA used 2×10^{-5}.

G Component Distribution in Top-K Edges

Figure 8 shows component-wise distribution of top-400 edges across four models and six datasets.

Normalised Frequency The raw composition in Figure 8 is dominated by the component types that are simply most numerous in the model (e.g., MLP and attention nodes), which makes it hard to see whether a component type is selected more often than its baseline abundance would predict. To remove this frequency bias, we report an architecture-normalised composition. Let f_c^{top} be the fraction of the top-400 EAP edges whose endpoint is of component type $c \in \{\text{attention, MLP, embedding, logit}\}$, and let f_c^{model} be the fraction of all components of type c in the full model. We define the *scaling factor*

$$r_c = \frac{f_c^{\text{top}}}{f_c^{\text{model}}},$$

which measures how strongly type c is over- or under-represented among the EAP-identified components relative to its natural abundance: $r_c > 1$ means type c is selected more often than chance, and $r_c < 1$ means it is under-selected. Figure 9 plots r_c directly for each component type (grouped along the x-axis) and task (bars). Under this normalisation, the contributions of MLP and especially logits are amplified, indicating that although these types are not the most numerous overall, they are disproportionately selected as causally important.

H Cross Task Performance Transferability

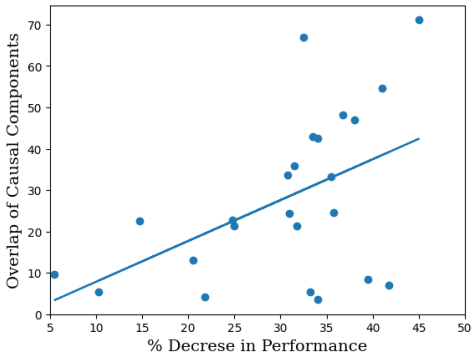


Figure 10: GPT2, $\rho = 0.5$

I Induction Head Detection Details

We detect induction heads with the synthetic repeated-sequence protocol of Olsson et al. (2022). We sample a random token sequence $X = [x_1, \ldots, x_T]$ of length $T = 50$, with token ids drawn uniformly from $[100, |V|)$ to avoid special and other reserved low-id tokens, and feed its twofold repetition $[X, X]$ of total length $2T = 100$ to a `HookedTransformer` (Nanda and Bloom, 2022), caching every layer's attention pattern. For each head we compute an *induction score*: the mean attention, averaged over query positions $i \in \{T, \ldots, 2T-2\}$ in the repeated half, that the head places on key position $i - T + 1$ —the token immediately following the previous occurrence of the current token. A head is labelled an induction head by a top-K rule on the sorted induction scores. Let K_{elbow} be the elbow of the sorted-score curve (the rank of maximum distance to the chord joining its first and last points), searched within the top 20% of ranks; let $K_{0.1}$ and $K_{0.3}$ be the numbers of heads scoring above 0.1 and 0.3, respectively. We set

$$K = \max\big(\min(K_{\text{elbow}},\ K_{0.1}),\ K_{0.3}\big),$$

i.e., the elbow cut, tightened by a minimum-score threshold of 0.1, but never dropping a head whose score exceeds 0.3 (a safety floor). The K highest-scoring heads are retained. The same protocol is applied to the pre-trained and the fine-tuned model.

J Validation of the correlation claim

§ 5.3 reports a positive correlation between the overlap in EAP-identified components and the degradation observed on Task 2 when the model

Similar Task Analysis									
Task Pair		**GPT2**		**Llama3.21B**		**Qwen2.05B**		**Llama-2-7B**	
Fine-tuning Task	Evaluation Task	Perf Δ	Overlap Δ	Perf Δ	Overlap Δ	Perf Δ	Overlap Δ	Perf Δ	Overlap Δ
Yelp	Yelp	34% ↑	100 %	33.81% ↑	100%	25.1% ↑	100%	91% ↑	100%
Yelp	SST-2	47% ↑	45.25%	52.2% ↑	39.75%	23.51% ↑	33.50%	94.5% ↑	16.75%
SST-2	SST-2	52.81%↑	100%	21.92% ↑	100%	31.77% ↑	100%	96.44%↑	100%
SST-2	Yelp	29.39% ↑	37.25%	8.41% ↑	40.5%	16.7% ↑	35.00 %	89.2% ↑	8.00 %
SQuAD	SQuAD	535% ↑	100%	417.54% ↑	100%	65.55% ↑	100%	81.23% ↑	100%
SQuAD	CoQA	216% ↑	37.00%	197.71% ↑	30.75%	41.16% ↑	35.50%	27.84% ↑	45.50%
CoQA	CoQA	287% ↑	100%	203% ↑	100%	106.59 % ↑	100%	5.95%↑	100%
CoQA	SQuAD	319% ↑	36.50%	312% ↑	31.75%	56.99% ↑	40.00 %	64.33% ↑	51.25%
KDE4	KDE4	441% ↑	100%	188% ↑	100%	119.90% ↑	100%	10.31% ↑	100%
KDE4	Tatoeba	206% ↑	47%	4.52% ↑	30.25%	4.13% ↑	37.50%	11.35% ↑	31.75%
Tatoeba	Tatoeba	384% ↑	100%	39.82%	100%	69.48% ↑	100%	24.53% ↑	100%
Tatoeba	KDE4	13.83% ↑	47.00%	76.84% ↑	32.50%	9.92% ↓	40.00%	9.33% ↑	29.25%
Dissimilar Task Analysis									
Yelp	SQuAD	67%	32.50%	81.71%	26.75%	9.50%	30.50%	0.59%	44.50%
Yelp	CoQA	9.64%	30.50%	52.81%	30.50%	38.18%	29.75%	2.8%	39.75%
Yelp	KDE4	54.61%	41.00%	82.92%	26.75%	38.97%	31.00%	8.20%	27.75%
Yelp	Tatoeba	22.63%	42.75%	87.3%	27%	26.85%	31.25%	0.08%	14.75%
SST-2	SQuAD	47%	36%	90.31%	32.25%	72.29%	36.25%	24%	7%
SST-2	CoQA	48.2%	36.75%	61.33%	26%	64.62%	36.25%	6.7%	14.75%
SST-2	KDE4	71.21%	42.00%	91.54%	29.75%	88.22%	39.25%	23.76%	19.75%
SST-2	Tatoeba	33.15%	41.50%	99.00%	34.25%	99.31%	39.75%	33.15%	19.75%
SQuAD	Yelp	83.91%	33.50%	8.41%	28.75%	58.31%	33.75%	2.2%	46.75%
SQuAD	SST-2	4.26%	31.75%	10.31%	31.50%	44.38%	35.50%	6.77%	6.25%
SQuAD	KDE4	24.63%	35.75%	6.91%	30.50%	41.82%	30.50%	66.52%	29.00 %
SQuAD	Tatoeba	21.35%	35.00%	3.81%	28.25%	37.07%	31.50%	87.81%	3.50%
CoQA	Yelp	0.31%	33.25%	1.81%	26.25%	4.21%	29.02%	4.51%	38.52%
CoQA	SST-2	3.63%	34.00%	4.67%	25.00%	3.78%	32.25%	1.03%	12.50%
CoQA	KDE4	35.83%	31.50%	12.63%	26.00%	31.46%	27.75%	25.86%	24.75%
CoQA	Tatoeba	21.33%	31.75%	7.91%	19.75%	35.75%	27.50%	21.44%	5.25%
KDE4	Yelp	3.31%	39.50%	5.82%	26.25%	68.12 %	33.00%	6.51%	30.25%
KDE4	SST-2	6.91%	41.75%	13.81%	34.50%	60.09%	41.50%	0.11 %	25.75%
KDE4	CoQA	33.72%	30.75%	16.01%	30.25%	4.38%	34.00%	4.91%	28.25%
KDE4	SQuAD	42.61%	34.00%	13.14%	31.25%	13.19%	35.50%	8.86%	28.50%
Tatoeba	Yelp	3.00%	40.50%	9.64%	26.25%	71.53%	35.25%	6.92%	23.00%
Tatoeba	SST-2	5.41%	40.25%	29.48%	31.50%	60.78%	41.75%	0.00%	14.00%
Tatoeba	CoQA	54.33%	31.00%	6.22%	25.50%	58.47%	32.00%	1.97%	14.75%
Tatoeba	SQuAD	25.61%	34.75%	26.24%	28.00%	91.00%	32.50%	6.56%	5.00%

Table 6: Cross-task Performance and Important Component Overlap Analysis for GPT-2 Small, Llama-3.2-1B, Qwen2-0.5B and Llama-2-7B.

is fine-tuned on Task 1. Because this evidence is correlational, it does not by itself establish that the shared components are responsible for the degradation. To test this mechanism directly, we perform an intervention: we identify the components that are shared between the two tasks, freeze them, and fine-tune only the remaining parameters on Task 1. If the overlapping components are indeed what drives the interference, protecting them during fine-tuning should recover part of the lost performance on Task 2.

Table 7 reports the change in Task 2 performance under this modified fine-tuning procedure, relative to standard full fine-tuning on Task 1. Across all model and task-pair combinations, freezing the overlapping components improves Task 2 performance, with gains ranging from 2.05% to 84.34%.

K Layer-wise Centred Kernel Alignment Similarity

The probing analysis in § 5.1 measures representational change by mapping intermediate hidden states through the model's output head (Equation 2). This quantity is therefore output-mediated: it captures how recoverable the task output is from a given layer, rather than how much the geometry of the representation itself has changed. Since the EAP objective is also defined in terms of logit differences, we complement the probing analysis with a measure that does not pass through the output head.

Specifically, we compute token-level hidden states Centred Kernel Alignment (CKA), a standard measure of similarity between layer-wise representations, between the pre-trained and fine-tuned models at every layer. A value of 1 indicates that the representation at that layer is unchanged by fine-tuning, so lower values correspond to larger representational change. Because the four models differ in depth, we plot CKA against relative depth (layer index divided by the total number of layers), which makes the curves directly comparable across models. Figure 11 shows the resulting curves for all models and datasets. Across all models and tasks, the most substantial layer-wise representational changes occur in the later layers (in the order

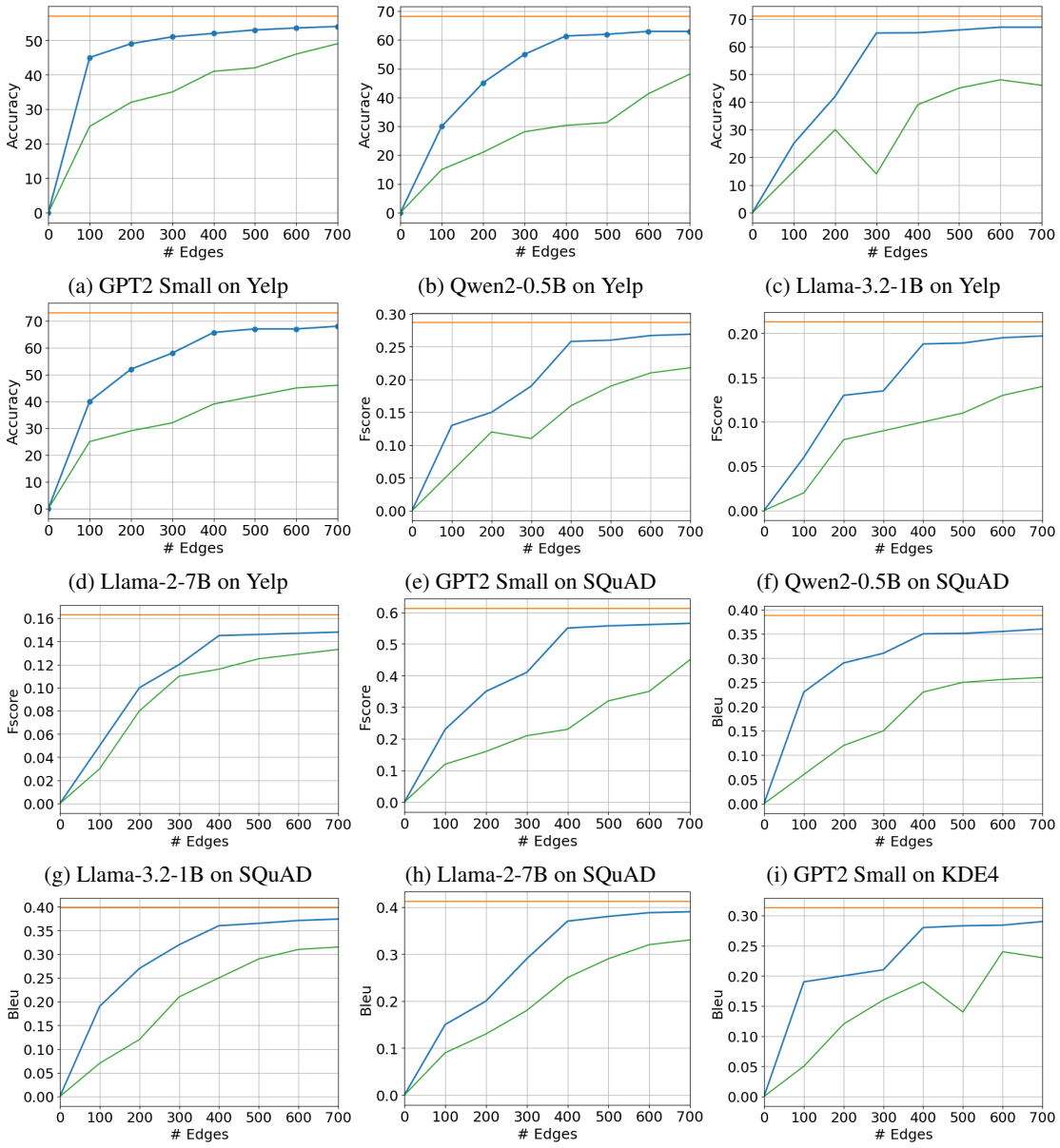
Model	FT task	Test task	Change
Llama-3.2-1B	MT	QA	↑ 3.54%
	QA	MT	↑ 45.74%
	Sent	MT	↑ 7.91%
	Sent	QA	↑ 41.93%
Qwen2-0.5B	QA	MT	↑ 84.34%
	MT	QA	↑ 44.47%
	MT	Sent	↑ 2.05%
	Sent	QA	↑ 83.82%
	Sent	MT	↑ 77.99%

Table 7: Effect of freezing the overlapping EAP-identified components during fine-tuning. The model is fine-tuned on Task 1 with the components shared between Task 1 and Task 2 held fixed, and then evaluated on Task 2. The last column reports the relative change in Task 2 performance compared to standard full fine-tuning on Task 1. Performance on Task 2 improves in every configuration.

of 0.7 to 0.9), specifically from approximately the 90th percentile layer to the final layer.

L More main results

This section contains the extended results of Figure 2 (see Figure 12 Figure 13), Figure 3 (see Figure 14), and Figure 4 (see Figure 10) for rest of models and datasets, which exhibit the same trends and confirm that our findings hold across all four model families.



(a) GPT2 Small on Yelp (b) Qwen2-0.5B on Yelp (c) Llama-3.2-1B on Yelp

(d) Llama-2-7B on Yelp (e) GPT2 Small on SQuAD (f) Qwen2-0.5B on SQuAD

(g) Llama-3.2-1B on SQuAD (h) Llama-2-7B on SQuAD (i) GPT2 Small on KDE4

(j) Qwen2-0.5B on KDE4 (k) Llama-3.2-1B on KDE4 (l) Llama-2-7B on KDE4

Figure 7: GPT-2 Small, Llama-3.2-1B, Qwen2-0.5B and Llama-2-7B performance with using only top-K edges obtained from EAP. The first row shows the performance on Yelp data, the second row shows the performance on SQuAD data, the third row shows the performance on KDE4 data. Orange line shows the performance of the finetuned model, blue line shows the performance with variation of top-k edges and green line shows the performance using randomly selected k edges.

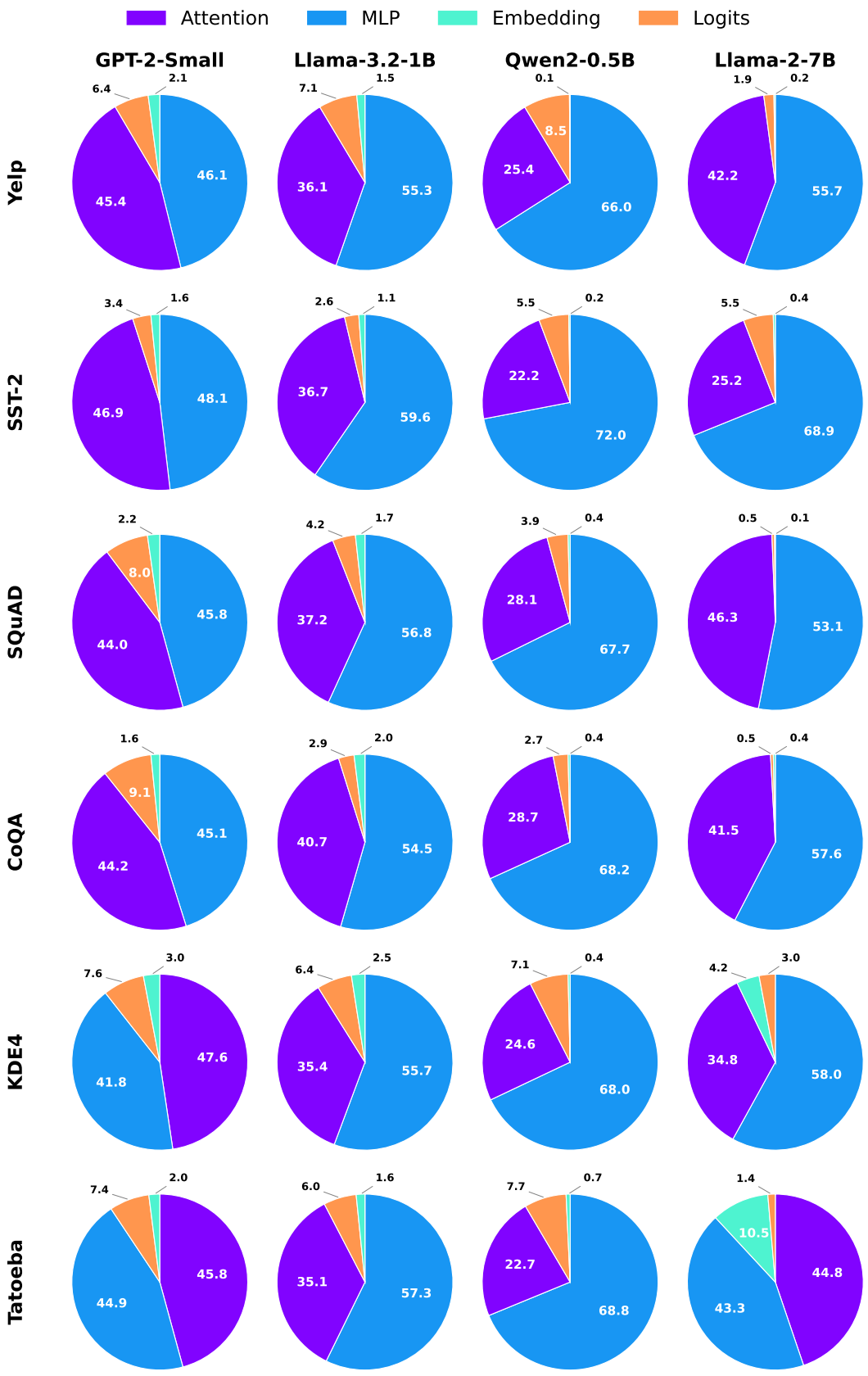
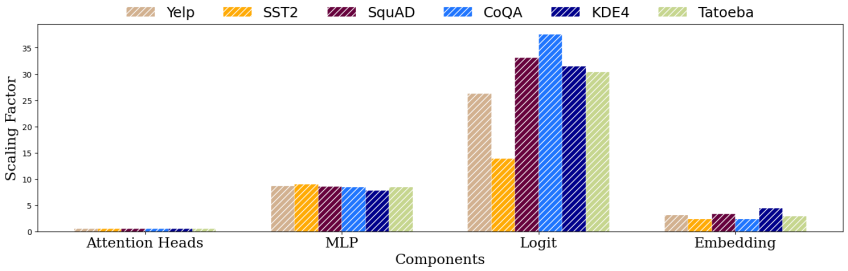
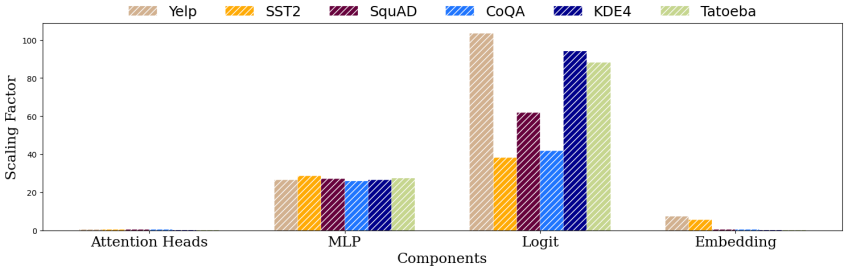


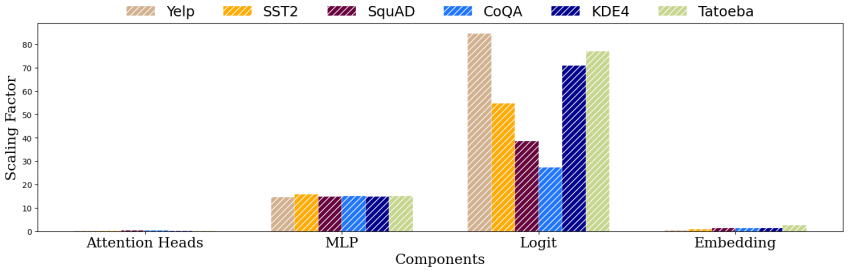
Figure 8: Important component distribution percentage over top-400 EAP edges, broken down by model(columns) and dataset(rows).The Llama-3.2-1B column on SST-2 / SQuAD / KDE4 reproduces the main-text Figure 5.



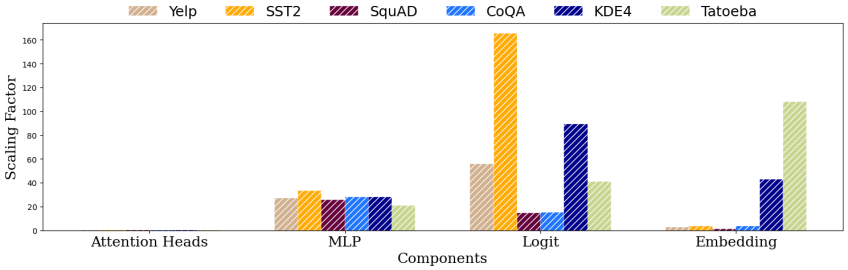
(a) GPT2 Small



(b) Llama-3.2-1B



(c) Qwen2-0.5B



(d) Llama-2-7B

Figure 9: Architecture-normalised component composition of the top-400 EAP edges. For each model (a-d), bars show the scaling factor $r_c = f_c^{\text{top}} / f_c^{\text{model}}$, i.e., the fraction of top-400 edges of component type c divided by the fraction of type c in the full model. Components are grouped along the x-axis (attention heads, MLP, logit, embedding) and the six bars per group correspond to the six tasks. A value above one indicates that the component type is selected more often than its baseline abundance would predict. See § G for the definition.

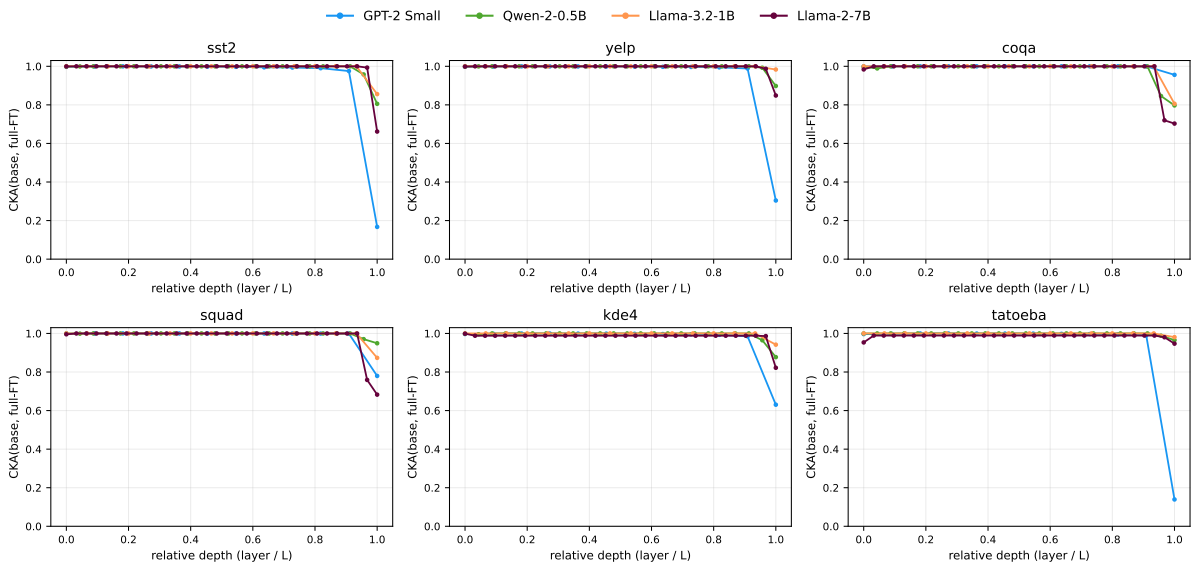
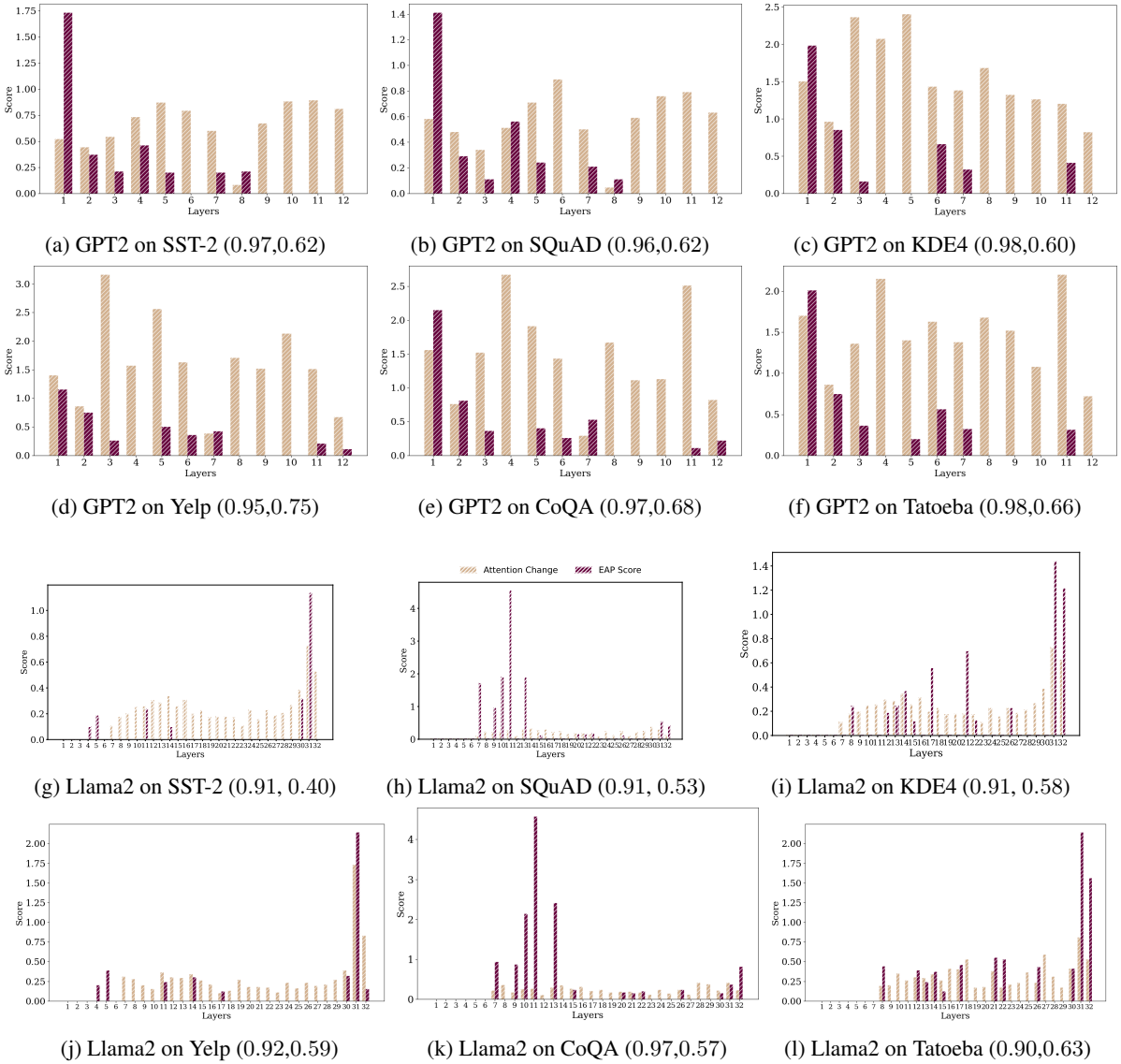


Figure 11: Token-level CKA between the pre-trained and the fine-tuned model at each layer, for the four models (lines) across the six datasets (panels). The x-axis is relative depth (layer index divided by the number of layers), which allows models of different depths to be compared on a common scale. A CKA of 1 means the representation at that layer is unchanged by fine-tuning; lower values indicate larger representational change.



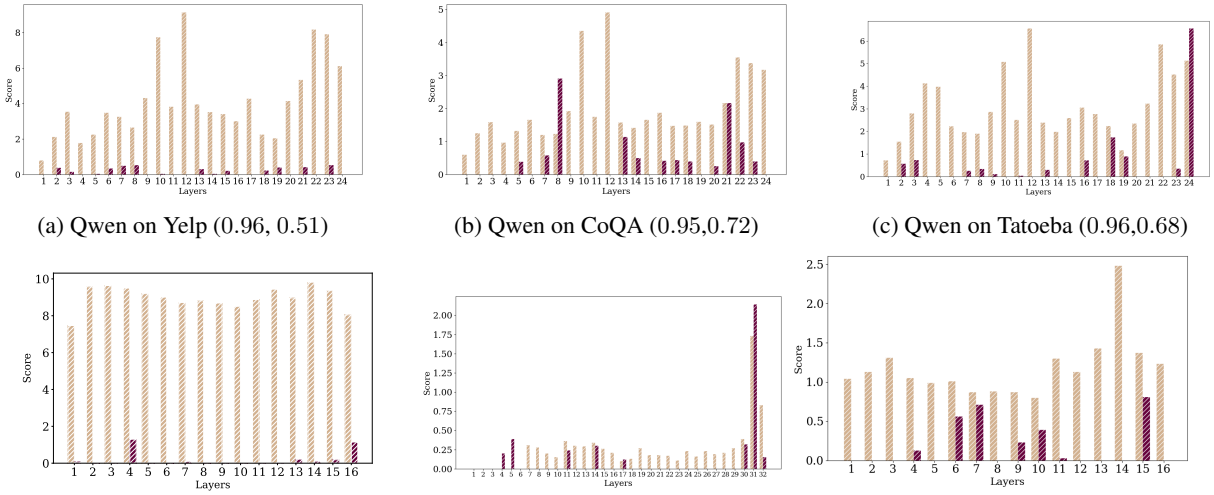
(a) GPT2 on SST-2 (0.97,0.62) (b) GPT2 on SQuAD (0.96,0.62) (c) GPT2 on KDE4 (0.98,0.60)

(d) GPT2 on Yelp (0.95,0.75) (e) GPT2 on CoQA (0.97,0.68) (f) GPT2 on Tatoeba (0.98,0.66)

(g) Llama2 on SST-2 (0.91, 0.40) (h) Llama2 on SQuAD (0.91, 0.53) (i) Llama2 on KDE4 (0.91, 0.58)

(j) Llama2 on Yelp (0.92,0.59) (k) Llama2 on CoQA (0.97,0.57) (l) Llama2 on Tatoeba (0.90,0.63)

Figure 12: Layer-wise distribution of average attention KL divergence and EAP scores for GPT-2 Small and Llama-2-7B across Sentiment Classification, Question Answering, and Machine Translation.



(a) Qwen on Yelp (0.96, 0.51) (b) Qwen on CoQA (0.95,0.72) (c) Qwen on Tatoeba (0.96,0.68)

(d) Llama3.2 on SST-2 (0.99, 0.65) (e) Llama3.2 on Yelp (1.00,0.48) (f) Llama3.2 on Tatoeba (0.98,0.61)

Figure 13: Layer-wise distribution of average attention KL divergence and EAP scores for Qwen2 and Llama-3.2-1B across Sentiment Classification (Yelp), Question Answering (CoQA), and Machine Translation (Tatoeba). Counterpart of Figure 2 .

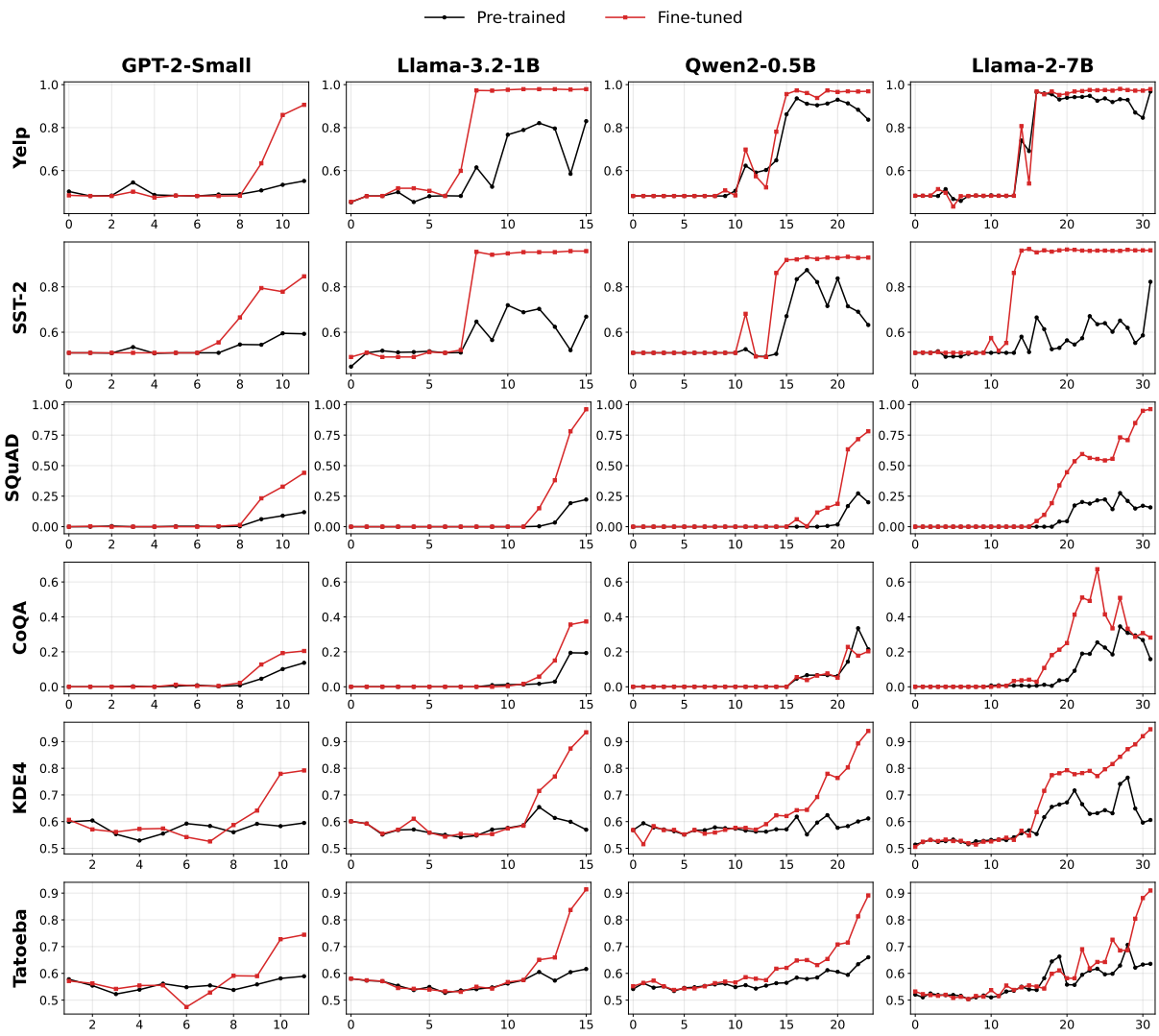


Figure 14: Layer-wise probing of pre-trained vs. fine-tuned models, full version of Figure 3. **Rows** are datasets (Yelp, SST-2, SQuAD, CoQA, KDE4, Tatoeba); **columns** are models.(GPT-2 Small, Llama-3.2-1B, Qwen2-0.5B, Llama-2-7B). Higher = more task-relevant information is already recoverable at that layer. (X-axis: layer index; Y-axis: binary accuracy for SST-2, SQuAD F1 Score for SQuAD, BERTScore for KDE4)