Automated Attention Pattern Discovery at Scale in Large Language Models

Jonathan Katzy[1], **Razvan Popescu**[1], **Erik Mekkes**[1], **Arie van Deursen**[1], **Maliheh Izadi**[1]

[1]Delft University of Technology
J.B.Katzy@TUDelft.nl

Abstract

Language models have scaled rapidly, yet methods for explaining their outputs are lagging behind. While initial methods focused on manual explanations, some works have started to automate the process. These methods however, are resource intensive and do not scale well with language model size. Previous studies demonstrated that specific attention heads consistently attend to similar tokens in a sequence and form recognizable patterns in the attention matrix. We present *AP-MAE*, a vision transformer-based masked auto-encoder to detect consistent clusters of these patterns across multiple language model invocations. Our experiments indicate that AP-MAE effectively reconstructs masked attention patterns. We show that these attention patterns are consistently distributed throughout the target models. Furthermore, we observe distinct behaviors in language models when generating correct versus incorrect predictions. Our results for the StarCoder 2 models indicate that our method can generalize across different models. This suggests that our auto-encoder can serve as a base model for further fine-tuning on other model families. Lastly, our method of tagging attention heads scales *linearly* with the number of attention heads in a language model and can function as a filtering step to enhance classical circuit discovery approaches.

Code — https://github.com/AISE-TUDelft/AP-MAE

Datasets — https://huggingface.co/datasets/LaughingLogits/Stackless_Java_V2

Models — https://huggingface.co/collections/LaughingLogits/ap-mae-models-66b27a73536bb1306d55c4c4

1 Introduction

With the widespread adoption of Large Language Models (LLMs) in recent years, understanding their outputs has become increasingly important. While advanced methods exist to explain which parts of an LLM contribute to specific outputs (Rai et al. 2024), these methods often struggle with scalability, fail to generalize across different models, or selectively focus on only a subset of components responsible for the output (Anwar et al. 2024).

A common finding in previous studies is the attribution of specific behaviors to particular heads within an LLM. We hypothesize that if certain heads consistently demonstrate specific behaviors, the output patterns of these attention heads will also share common features. The attention output indicates the degree to which the model attends to a particular token in the input sequence when generating its output. This has been used to explain the reasoning behind transformer models' decisions (Hao et al. 2021; Vig 2019).

In previous studies, the term "head" has been used to describe both the attention head in the model and its observed behavior. In this work, we distinguish between the two. Here, "head" specifically refers to the sub-module within the transformer architecture, while "attention patterns" refer to the behavior or output of a head. This distinction is important because a single head can display different behaviors depending on the inputs, and similar behaviors can be exhibited by multiple heads.

Our research focuses on automatically detecting attention patterns in LLMs at scale. We address this by training a Vision Transformer-based Masked Auto-Encoder (ViT-MAE) (Dosovitskiy et al. 2021; He et al. 2022), which we introduce as *AP-MAE* (Attention Pattern - Masked Auto-Encoder). AP-MAE generates intermediate representations that we use to cluster similar patterns. Then, we analyze the distribution of these patterns within LLMs.

This approach helps us explore two key research questions: *(i)* Do LLMs produce recognizable patterns in their attention outputs? *(ii)* How are these attention patterns distributed throughout the model?

Additionally, we address some of the limitations of existing mechanistic interpretability research. A key issue with current methods is their reliance on multiple invocations of an LLM to identify circuits (Conmy et al. 2023; Wang et al. 2023). Our approach, once trained, allows for tagging relevant attention heads with a single model invocation, achieving scalability that increases *linearly* with the number of *heads* in the targeted LLM.

Moreover, the datasets used for evaluation have faced criticism. In large-scale assessments of LLMs, evaluating on data from similar but distinct datasets can significantly alter data representation (Bolukbasi et al. 2021). For smaller investigations, prompt templates are often used to elicit desired behaviors (Wang et al. 2023). These limitations may lead to spurious correlations or incorrect conclusions. To address this, our approach closely mirrors the dataset used to

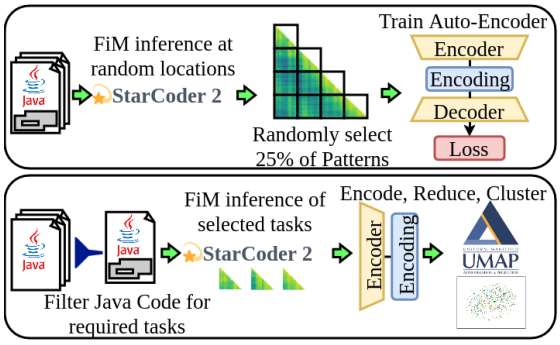


Figure 1: Overview of the methodology; top: Training procedure, bottom: Clustering procedure

train the targeted LLMs without any overlap. Instead of creating specific prompts, we extract the necessary completions directly from the dataset.

We conducted this investigation in two steps. First, we trained the AP-MAE model using attention patterns gathered from the target LLM when masking random spans of the dataset. We selected the StarCoder2 (hereafter called *SC2* due to brevity) models (Lozhkov et al. 2024) as our target LLMs due to their open training data, which allows for replication on an unseen dataset, and because they were trained on source code. We prefer code for its structured nature, which enables us to query a wide range of scenarios. Among programming languages, we choose Java because of its verbosity and popularity. In the second stage, we select specific scenarios from the code that have been linked to certain behaviors and encode the attention patterns observed when prompting the SC2 models. These encodings are then used to cluster the patterns and analyze the models' behavior on a large scale. Figure 1 provides an overview of the methodology. We make the following contributions:

- Releasing StackLessV2, a companion dataset for The Stack v2.
- Releasing AP-MAE models for SC2 attention patterns.
- Propose a clustering approach to classify and detect attention patterns using AP-MAE encodings.
- Provide empirical evidence that patterns are consistently distributed in LLMs at scale.
- Demonstrate that the locations where a pattern appears differ distinctly depending on whether the inference is correct or incorrect.

2 Background and Related Work

2.1 Auto-Encoders

Auto-Encoders are a subset of neural models that focus on either compressing or extracting features from a sample. When working with images, vision transformers (Dosovitskiy et al. 2021), can be trained to learn representations of 2 dimensional data by masking up to 70% of an image and having the model reconstruct the original image (He et al. 2022). The encodings learned by these models can then be used for object detection, image segmentation, and classification (clustering).

2.2 Mechanistic Interpretability

Mechanistic interpretability focuses on explaining the outputs of LLMs by analyzing their internal components. The methods used to evaluate these internal mechanisms can be grouped into two main categories: feature extraction and circuit discovery.

Feature Extraction Feature extraction involves utilizing sparse auto-encoders to identify abstract features that contribute to model predictions. This process can be applied at various points within the transformer architecture, such as the neurons in the MLP (Bricken et al. 2023) or attention scores (Kissane et al. 2024). Once the features are extracted, experts label them to interpret their meaning (Bricken et al. 2023; Kissane et al. 2024; Zou et al. 2023). However, this labeling step may impose human concepts on the features, potentially obscuring more complex patterns or reinforcing spurious correlations.

Circuit Discovery Circuit discovery, unlike feature extraction, emphasizes understanding how components interact within a network. A circuit refers to the path that information follows and is typically analyzed at the attention head level. So far, various circuits have been identified, including the 'Indirect Object Identification' circuit (Wang et al. 2023) and the 'Docstring' circuit (Heimersheim and Janiak 2023). These circuits consist of attention heads, each performing specific tasks, such as 'Induction Heads' (Olsson et al. 2022), 'Previous Token Heads' (Elhage et al. 2021), 'Duplicate Token Heads' (Krzyzanowski et al. 2024), and 'Name Mover Heads' (Wang et al. 2023).

Early methods for identifying circuits relied on manual analysis, but recent approaches have automated this process (Conmy et al. 2023). They can find previously discovered manual circuits as well as identify new computational ones. However, they require pruning transformer network heads to isolate the smallest possible circuit. This greedy strategy demands multiple LLM inferences, and can overlook secondary circuits that the model can use to generate correct predictions (Wang et al. 2023; Anwar et al. 2024). To address the inefficiency of multiple inferences, researchers have called for developing heuristics to reduce the algorithmic search space or automating the process in other ways (Anwar et al. 2024; Räuker et al. 2023).

3 The Dataset - StackLessV2 Java

To train our encoder model, we needed attention patterns from a dataset similar to the one used for training SC2 models, which were trained on "The Stack v2" dataset. The Stack v2 comprises all GitHub code files released under a permissive license or without a license. To create a comparable dataset, we scraped GitHub, specifically targeting repositories with non-permissive licenses to ensure that the code would not overlap with The Stack v2. Since we are not training a generative model, concerns about original author attributions in our model outputs are minimal (Al-Kaswan and

Step	Number of Files
Raw data	7.80 M
Auto-generated	0.04 M
Exact-deduplication	5.63 M
Near-deduplication	0.80 M
Final	1.33 M

Table 1: Number of Java files in each step.

Izadi 2023). We named our dataset "StackLessV2", to highlight its role as a companion dataset to The Stack v2.

Collection To create the dataset, we scraped $10,500$ public repositories using the GitHub API. We aim to include $10,000$ repositories in the dataset and select an extra 500 to account for repositories being removed between collection and downloading. We select the licenses GPL-2.0, GPL-3.0, and AGPL-3.0, and scraped files up until April 2024. We order the results by the number of stars in descending order, and extracted Java files from the repositories using their file extension, i.e., `.java`.

Cleaning After collecting the data from GitHub, we performed data cleaning. We excluded Java files larger than 50 MB and those with fewer than 10 words. According to the procedure used to generate The Stack v2 (Lozhkov et al. 2024), we then removed auto-generated files by searching for specific phrases in the first 5 lines of each file.

Deduplication Selecting repositories based solely on licenses is often insufficient for creating distinct datasets (Katzy et al. 2024). To prevent data leakage and ensure uniqueness, we deduplicated our dataset against The Stack v2. We performed exact deduplication by comparing SHA-256 hashes and removed near duplicates with a Jaccard similarity above 0.7. We chose this standard threshold value as it has been shown to be robust for duplicate detection (Allamanis 2019). Moreover, this value reduces the risk of false negatives, even if it increases the possibility of false positives. Table 1 summarizes the number of files removed at each step and the final dataset size.

4 AP-MAE

To train AP-MAE, we model the patterns as grayscale images and base our model on the original ViT-L architecture (Dosovitskiy et al. 2021). We apply a novel scaling method to the attention patterns to allow the training to converge. Figure 2 provides an overview of the AP-MAE model.

Architecture AP-MAE is based on the ViT-L architecture with minor changes. We reduce the hidden size from 768 to 512 dimensions and accordingly scale the MLP down to 2048 parameters. We also remove the top right triangle of the data as this is masked in decoder-only attention patterns. For patches on the matrix's diagonal, we pad the masked values above the diagonal. Finally, we scale the attention patterns by taking the natural logarithm. This step is essential in allowing the model to fit to attention patterns, as demonstrated in Section 4. Table 2 presents an overview of the setup.

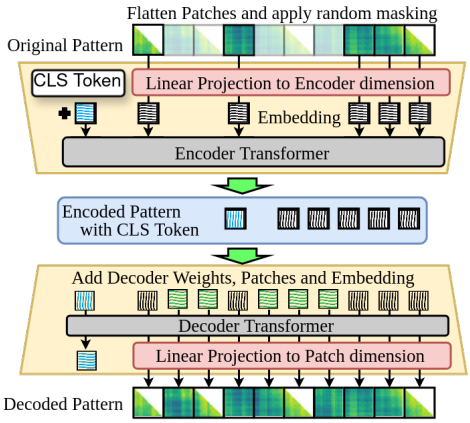


Figure 2: Overview of the model design

Model Inputs		Encoder		Decoder	
Pattern Size	256	Layers	24	Layers	8
Patch Size	32	Dim	512	Dim	512
Mask Ratio	0.5	Heads	16	Heads	8
Batch Size	480	Head Dim	64	Head Dim	64
LR	0.00144	MLP	2048	MLP	2048

Table 2: Training and architecture parameters for AP-MAE.

Training Procedure We train three separate models. Each model is fit on attention patterns from SC2 3B, 7B, or 15B. The attention patterns used for training are generated by prompting the target model with data from the StackLessV2 dataset. We randomly mask a span of between 3-10 tokens in the file. We also set the maximum length of the input to 256 tokens and truncate the left and right contexts.

For each target model prompt, we sub-sample the generated attention patterns to 25% of the heads in each layer. This allows a larger variety of files to be seen, as each target model inference generates between 720 and 1920 attention patterns. In Figure 3, we show the initial attention pattern on the left. Then we display the masking and the reconstructed parts of the masked pattern. Finally, we present the combination of the original and reconstructed patterns.

Training Setup We train the models on eight A100 GPUs with a local batch size of 60 attention patterns. We train for $150,000$ batches using 72M attention patterns. The total training time is less than 100 GPU hours on our institution's cluster. Although we used eight A100 GPUs, all target models and AP-MAE models combined fit on one A100, making it possible to train on smaller servers. The limiting factor is the LLMs size, as the AP-MAE encoder has only 101M parameters and the decoder has 25M parameters. We use the AdamW optimizer with a weight decay of 0.05 and a cosine annealing scheduler initialized with a global Learning Rate (LR) of 1.44×10^{-3}. We linearly upscale from the base LR of 1.5×10^{-4} for a batch size of 50 used by ViT-MAE.

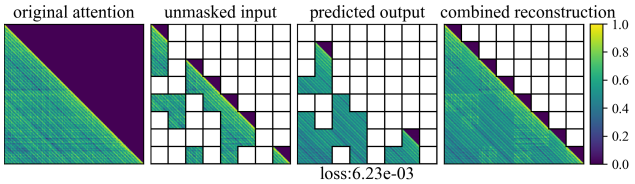


Figure 3: Attention pattern reconstruction after training on log normalized attention data

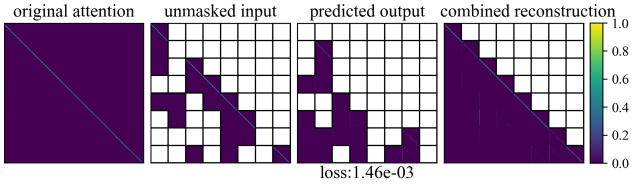


Figure 4: Attention pattern reconstruction after training on raw attention data

Ablation Study The main difference between our approach and the ViT-MAE models is the logarithmic scaling of the input. To show the necessity of this step, we train an identical model, without scaling. In Figure 4, we show an unscaled attention pattern. In this image, it is difficult to see the reconstruction. In Figure 5, we show the same pattern but scale the color gradient logarithmically to visualize the reconstruction. We see that the patches that were masked are corrupted. We compare this with the output of a model that has been trained on scaled attention patterns, by prompting it with the same attention pattern in Figure 3. We see that when using logarithmic scaling, major patterns are reconstructed. While we cannot compare loss values directly, this shows the necessity for the logarithmic scaling of the attention patterns when training and evaluating the AP-MAE models.

Generalizability To investigate AP-MAE's ability to reconstruct attention patterns from other models, we cross-evaluate the models on a test set, for all combinations of SC2 target models. We give an overview of the results in Table 3. Based on this table, we see that evaluating the encoder models on attention patterns from other target models results in a loss within one standard deviation of each other.

5 Clustering Attention Patterns

When generating the samples for clustering, we select scenarios and features that have been shown to elicit some type of behavior in previous works (Conmy et al. 2023; Kissane et al. 2024; Bricken et al. 2023; Izadi, Gismondi, and Gousios 2022). Based on these studies, we select the following tasks to extract from the code corpus: 1. Closing braces, 2. End of Line (EOL) prediction, 3. Boolean Literals, 4. String Literals, 5. Numeric Literals, 6. Identifiers, 7. Boolean operators, 8. Numeric Operators, 9. Assignment Operators, 10. Random masking. For clustering, we use all attention patterns generated when prompting the model $1,000$ times per task. 500 prompts produced correct inferences of the target model, and 500 belong to incorrect

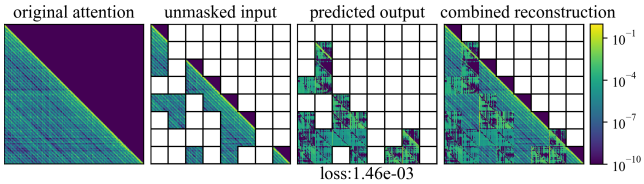


Figure 5: Attention pattern reconstruction after training on raw attention data. Pixel values are scaled for visualization

		SC2 Evaluated		
		3B Loss $\times 10^{-2}$	7B Loss $\times 10^{-2}$	15B Loss $\times 10^{-2}$
SC2 Trained	3B	1.39 ± 0.41	1.44 ± 0.44	1.64 ± 0.56
	7B	1.50 ± 0.43	1.41 ± 0.41	1.60 ± 0.41
	15B	1.80 ± 0.47	1.75 ± 0.44	1.47 ± 0.39

Table 3: Cross-evaluation of AP-MAE

inferences. We plot the first two dimensions of the raw encodings in Figure 6-A. We color the encodings by the head that generated the pattern. We observe that on the right of the plot, there are discrete clusters of patterns from a given head, while on the left the embeddings are mixed together.

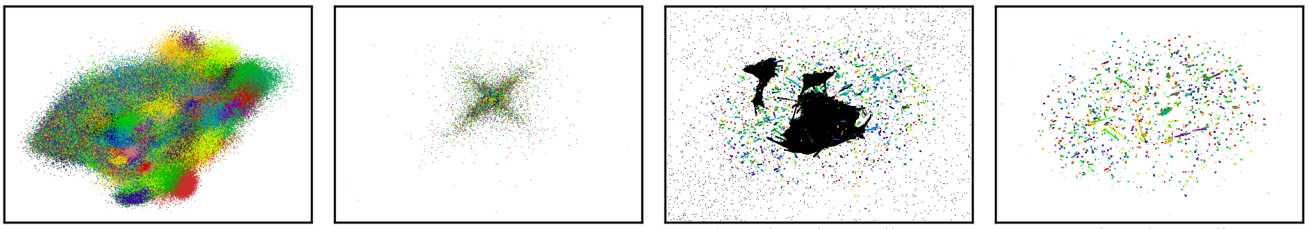
5.1 UMAP

Before clustering, we reduce the dimensionality of the data. This has been shown to boost the performance of downstream clustering algorithms (Allaoui, Kherfi, and Cheriet 2020) and reduces the resources required to cluster the 19.2 million samples. We choose UMAP for dimensionality reduction as it is a nonlinear method, and once fitted, can transform new samples to the lower dimensional space¿ This makes it effective for use in a pipeline (McInnes, Healy, and Melville 2018). We use the hyper-parameters; 15 nearest neighbors, and minimum distance between points of 0.1. For the UMAP algorithm and later the trustworthiness score, we use the Manhattan distance for its performance in higher dimensions (Aggarwal, Hinneburg, and Keim 2001). In Figure 6-B, we observe the formation of individual clusters. A large mass appears near the center of the plot, with clusters becoming smaller and more dispersed as they move away from the center.

To find the optimal number of target dimensions, we use the trustworthiness score (Venna and Kaski 2001). In this metric, we compute the $1,000$ nearest neighbors of a point before and after the transformation, then measure the overlap between these two sets. We conducted five trials of the UMAP algorithm and generated ten subsets of $1,000,000$ samples each, as using the entire dataset was not feasible. The results, shown in Figure 7, indicate that a four-dimensional space provides the best balance between dimensionality reduction and preserving local neighborhoods.

5.2 HDBSCAN

We use the HDBSCAN algorithm (Campello, Moulavi, and Sander 2013) for clustering due to its compatibility with UMAP and its ability to handle noisy data (Allaoui, Kherfi,



A) Raw encodings Colored by head number

B) Reduced encodings Colored by head number

C) Reduced encodings Colored by cluster Zoomed to (15, -15)

D) Reduced encodings Only clustered samples Zoomed to (15, -15)

Figure 6: Clustering procedure after every step for the SC2 15B model

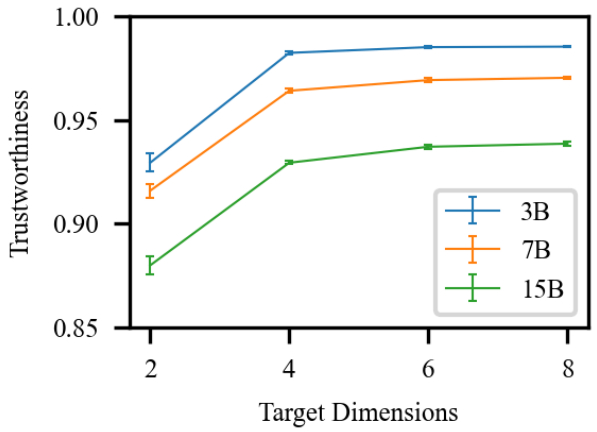


Figure 7: Trustworthiness score for different dimensions

and Cheriet 2020). HDBSCAN has also been successfully applied to clustering word embeddings (Asyaky and Mandala 2021), a task closely related to ours. Additionally, like UMAP, HDBSCAN can classify new points after being fit to an initial dataset.

We applied the HDBSCAN algorithm with a minimum separation of 0.5. Clusters were excluded if they had an average intra-cluster distance greater than 1 or if their total size was below 100 for SC2 3B, 160 for SC2 7B, and 267 for SC2 15B. These thresholds account for differences in the number of attention heads capable of generating an attention pattern in each model. We display the clustering results in Figure 6-C and D, with a focus on the central region to highlight the clusters. Numerous distinct clusters were identified. The large mass in the center, was filtered out due to exceeding the intra-cluster distance limit. These filtered points correspond to the left side of Figure 6-A. Additionally, small clusters on the outskirts were removed based on the minimum size criteria.

Model	Number of Clusters	Clustered Patterns	Total Patterns	Percentage Clustered
3B	197	1,095,148	7,200,000	15.2%
7B	301	1,667,434	11,520,000	14.5%
15B	797	3,175,823	19,200,000	16.5%

Table 4: Attention pattern clustering statistics

6 Results

6.1 Identifiable Patterns

Table 4 presents the number of clusters identified after running all 10,000 inferences for each target model. Here, each cluster corresponds to a unique pattern generated in the LLMs attention heads. We also visualize the distinctions between clusters by giving an overview of nine discrete clusters in Figure 8. To show the variation within a cluster, we select and display the most distant points on the convex hull of a cluster in Figure 9.

In the visualizations, distinct patterns emerge, indicated by horizontal, vertical, or diagonal lines that differentiate between clusters. Notably, the first and fifth patterns in Figure 8 exhibit different regions within one pattern. In examining intra-cluster variations, it is evident that while the overall pattern consists of multiple horizontal lines, these lines can shift to different positions.

Based on the results in Table 4, we note an increase in the number of clusters detected with model size. Additionally, between 14.5% and 16.5% of all collected patterns were assigned to a cluster. Most of the patterns that are not assigned a cluster are part of the large central mass depicted in Figure 6-C. Our study is the first to evaluate attention patterns in LLMs at this scale. We compared our findings with other circuit discovery studies, which identified that 17 out of 144 heads (11.8%) in GPT-2 were assigned to a primary circuit (Conmy et al. 2023). This proportion aligns with our results. However, it is important to note that the approach taken by Conmy may underestimate the number of relevant attention heads by not accounting for secondary and tertiary circuits (Anwar et al. 2024).

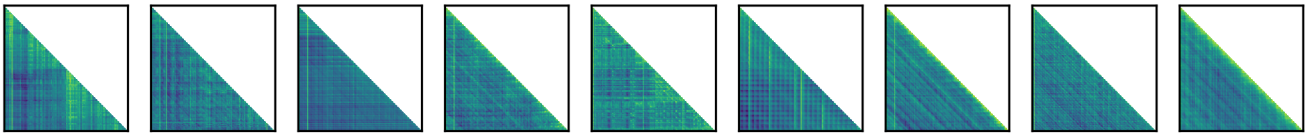


Figure 8: Examples of patterns found by clustering

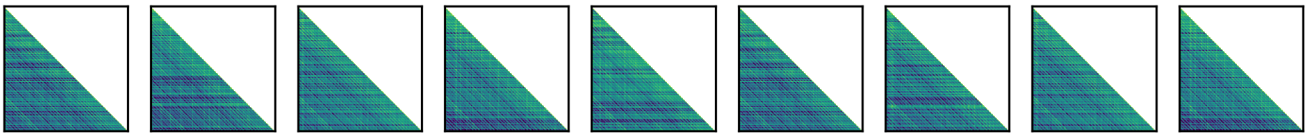


Figure 9: Attention patterns found within a single cluster

6.2 Pattern Distribution

To analyze the distribution of patterns in an LLM, we present a numerical summary of unique pattern locations across various tasks. We define a pattern location by combining the cluster ID and the location of the head that generated the pattern. Table 5 provides a detailed summary, including a breakdown between correct and incorrect predictions. We document the pattern locations for all models and tasks and note the total number of pattern locations utilized during $1,000$ inferences for each task. We also determine the number of pattern locations unique to each task, which is significantly lower than the total number of patterns locations. This suggests a substantial reuse of patterns and heads across tasks. Figure 10 depicts the distribution of patterns in a single prediction, and aggregates all 10,000 predictions to show the percentage of all recognizable generated patterns in a head and the number of distinct patterns generated in a head. Figure 10-B supports the findings in Table 5, showing that many parts of the model rarely produce a noticeable pattern.

We also examine the overlap between pattern locations associated with correct and incorrect predictions. Our findings show minimal overlap between these sets of pattern locations. We postulate that this minimal overlap is due to the model generating patterns from different tasks or languages, leading to erroneous predictions. This provides preliminary evidence of potential 'destructive interference' in LLMs. Additionally, it suggests that accurate prediction determination may be possible with sufficient prior knowledge of the specific task, although further investigation is required.

Analyzing the distribution of patterns within each model, we calculate the limit of potential pattern locations by multiplying the cluster counts (Table 4) by the number of heads in each model. For the 3B model, this gives us a total of $141,840$ pattern locations, for the 7B model $346,752$ and for the 15B model $1,530,240$. Comparing the values in Table 5 with the maximum values highlights the uniformity of the placement of the patterns within the LLM. When looking at the random masking task (the most diverse), we see that the target LLMs use 0.58%, 0.52%, and 0.24% of possible pattern locations in the 3B, 7B, and 15B models, respectively.

We compare these findings with previous studies. For GPT-2, heads are categorized as early, middle, and late heads (Kissane et al. 2024) and assigned specific behavior such as syntax matching and tense verification across different layers. We focus on the distribution of patterns, and see a similar segmentation emerge in Figure 10-B and C. In the early layers (0 and 1) of the SC2 15B model, a high percentage of patterns are clustered and a low number of distinct clusters are identified. In the middle layers 2 and 15, some heads rarely generate a recognizable pattern, and we see a high number of distinct patterns per head. In the late layers, few heads generate recognizable patterns, and the number of distinct patterns decreases. We believe the sparsity of heads that generate recognizable patterns is due to SC2 15B training including more than 600 programming languages, and we only investigate Java. Additionally, the number of attention locations identified in the random masking process, from Table 5, suggests that there are unexplored aspects of Java.

7 Discussion

The main finding of our study is that attention patterns in LLMs are both identifiable and consistently distributed. This discovery has significant implications for both pruning and circuit discovery in LLMs. By *automatically* identifying essential heads for specific tasks, we can prune models by removing non-essential heads, optimizing performance. Similarly, AP-MAE helps exclude certain heads when constructing computational circuits. As demonstrated in Figure 10, this method substantially narrows the search space.

Additionally, the minimal overlap in unique pattern locations between correct and incorrect predictions suggests that analyzing pattern locations on a large scale could reveal whether incorrect predictions result from the model generating attention patterns associated with a different task or language it was trained on. This could serve as a filter within a prediction pipeline to determine whether to return a result.

Finally, the experiments on generalizability described in

SC2 Size	Count	Assignment Operators	Boolean Literals	Boolean Operators	Closing Bracket	EOL	Identifier	Math Operators	Numeric Literals	String Literals	Random
3B	Total	422	422	403	544	643	564	652	613	470	820
	Unique	26	41	25	70	112	88	130	115	47	242
	Correct	8	16	7	48	32	44	31	21	38	144
	Incorrect	18	25	18	22	77	42	98	93	8	96
	Overlap	-	-	-	-	3	2	1	1	1	2
7B	Total	765	749	720	946	972	1,051	1,035	1,060	723	1,847
	Unique	110	72	49	121	158	171	181	221	55	779
	Correct	65	29	25	43	71	74	98	164	26	255
	Incorrect	44	43	23	76	81	93	76	55	28	507
	Overlap	1	-	1	2	6	4	7	2	1	17
15B	Total	1,928	1,921	1,788	2,279	2,488	2,462	2,831	2,561	2,542	3,630
	Unique	195	237	142	287	436	438	717	469	478	1,300
	Correct	53	152	92	144	212	277	175	285	350	434
	Incorrect	139	83	47	139	213	152	530	168	121	855
	Overlap	3	2	3	4	11	9	12	16	7	11

Table 5: Number of unique attention pattern locations per task

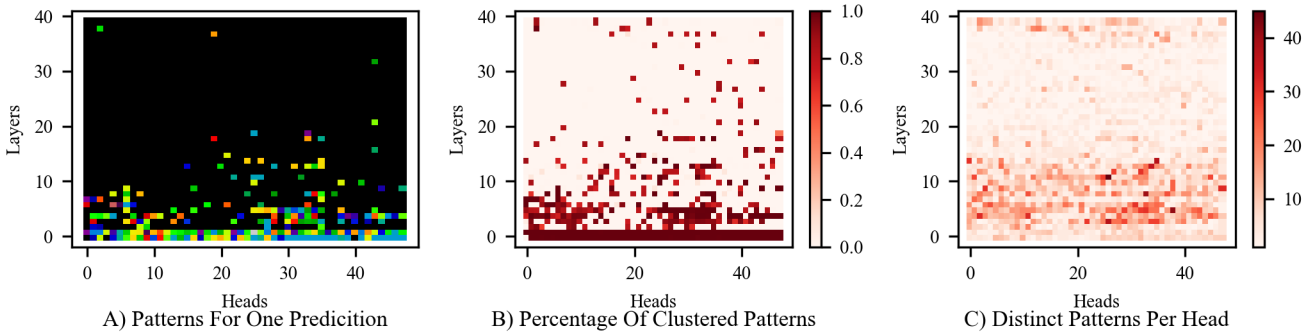


Figure 10: Overview of the distribution of attention patterns in SC2 15B, A) shows the attention patterns present in a single inference, B) and C) summarize all 10,000 inferences made for this investigation, where B) shows the percentage of patterns generated in a head that belong to any cluster, and C) shows the number of distinct patterns that were detected in every head.

Section 4 demonstrates that variations in the target model minimally impact the capacity of AP-MAE to reconstruct attention patterns. This suggests a certain universality of attention patterns across LLMs and could make it possible to train AP-MAE models that work for all target LLMs.

Limitations Having demonstrated the strengths of our method, there are some limitations that warrant future investigation. Firstly, the current ViT does not scale well with image size. This issue impacts the AP-MAE model, as the attention patterns depend on the length of the input. Future work could explore the use of transformers optimized for higher-resolution images (Yao et al. 2024) or enhancing efficiency (Liu et al. 2023). Secondly, our work draws inspiration from circuit discovery research to explore whether patterns manifest consistently on a larger scale; however, it does not involve actual circuit discovery. A logical next step would be to connect these findings with those from circuit discovery studies. Thirdly, we have demonstrated that our method is effective for decoder-only transformers. We anticipate similar success with encoder models, which we propose as a direction for future research. Finally, our investigation covered a subset of Java. Future research should expand this evaluation to encompass all aspects of Java and explore differences between various programming languages.

8 Conclusion

We demonstrated that LLMs consistently generate identifiable attention patterns located in specific heads. To facilitate this study, we developed the AP-MAE models and have made them publicly available alongside the StackLessV2 Java dataset. To our knowledge, this investigation is the largest of its kind, both in terms of the number of samples and the size of the models examined. Our findings highlight the scalability of our method and its applicability on a large scale. Lastly, we propose applications for our model that reduce computations during inference by pruning and optimizing the search space for computational circuits within LLMs.

References

Aggarwal, C. C.; Hinneburg, A.; and Keim, D. A. 2001. On the surprising behavior of distance metrics in high dimensional space. In *Database theory—ICDT 2001: 8th international conference London, UK, January 4–6, 2001 proceedings 8*, 420–434. Springer.

Al-Kaswan, A.; and Izadi, M. 2023. The (ab)use of Open Source Code to Train Large Language Models. In *2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE)*, 9–10.

Allamanis, M. 2019. The adverse effects of code duplication in machine learning models of code. In *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2019, 143–153. New York, NY, USA: Association for Computing Machinery. ISBN 9781450369954.

Allaoui, M.; Kherfi, M. L.; and Cheriet, A. 2020. Considerably Improving Clustering Algorithms Using UMAP Dimensionality Reduction Technique: A Comparative Study. In El Moataz, A.; Mammass, D.; Mansouri, A.; and Nouboud, F., eds., *Image and Signal Processing*, 317–325. Cham: Springer International Publishing. ISBN 978-3-030-51935-3.

Anwar, U.; Saparov, A.; Rando, J.; Paleka, D.; Turpin, M.; Hase, P.; Lubana, E. S.; Jenner, E.; Casper, S.; Sourbut, O.; et al. 2024. Foundational Challenges in Assuring Alignment and Safety of Large Language Models. *Submitted to Transactions on Machine Learning Research*. Under review.

Asyaky, M. S.; and Mandala, R. 2021. Improving the Performance of HDBSCAN on Short Text Clustering by Using Word Embedding and UMAP. In *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 1–6.

Bolukbasi, T.; Pearce, A.; Yuan, A.; Coenen, A.; Reif, E.; Viégas, F.; and Wattenberg, M. 2021. An Interpretability Illusion for BERT. arXiv:2104.07143.

Bricken, T.; Templeton, A.; Batson, J.; Chen, B.; Jermyn, A.; Conerly, T.; Turner, N.; Anil, C.; Denison, C.; Askell, A.; Lasenby, R.; Wu, Y.; Kravec, S.; Schiefer, N.; Maxwell, T.; Joseph, N.; Hatfield-Dodds, Z.; Tamkin, A.; Nguyen, K.; McLean, B.; Burke, J. E.; Hume, T.; Carter, S.; Henighan, T.; and Olah, C. 2023. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Campello, R. J. G. B.; Moulavi, D.; and Sander, J. 2013. Density-Based Clustering Based on Hierarchical Density Estimates. In Pei, J.; Tseng, V. S.; Cao, L.; Motoda, H.; and Xu, G., eds., *Advances in Knowledge Discovery and Data Mining*, 160–172. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.

Conmy, A.; Mavor-Parker, A.; Lynch, A.; Heimersheim, S.; and Garriga-Alonso, A. 2023. Towards Automated Circuit Discovery for Mechanistic Interpretability. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 16318–16352. Curran Associates, Inc.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Elhage, N.; Nanda, N.; Olsson, C.; Henighan, T.; Joseph, N.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; Conerly, T.; DasSarma, N.; Drain, D.; Ganguli, D.; Hatfield-Dodds, Z.; Hernandez, D.; Jones, A.; Kernion, J.; Lovitt, L.; Ndousse, K.; Amodei, D.; Brown, T.; Clark, J.; Kaplan, J.; McCandlish, S.; and Olah, C. 2021. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Hao, Y.; Dong, L.; Wei, F.; and Xu, K. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 12963–12971.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16000–16009.

Heimersheim, S.; and Janiak, J. 2023. A circuit for Python docstrings in a 4-layer attention-only transformer. *AlignmentForum*.

Izadi, M.; Gismondi, R.; and Gousios, G. 2022. CodeFill: multi-token code completion by jointly learning from structure and naming sequences. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, 401–412. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392211.

Katzy, J.; Popescu, R.; Van Deursen, A.; and Izadi, M. 2024. An Exploratory Investigation into Code License Infringements in Large Language Model Training Datasets. In *Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering*, FORGE '24, 74–85. New York, NY, USA: Association for Computing Machinery. ISBN 9798400706097.

Kissane, C.; Krzyzanowski, R.; Bloom, J. I.; Conmy, A.; and Nanda, N. 2024. Interpreting Attention Layer Outputs with Sparse Autoencoders. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Krzyzanowski, R.; Kissane, C.; Conmy, A.; and Nanda, N. 2024. We Inspected Every Head in GPT-2 Small Using SAEs So You Don't Have To. Alignment Forum.

Liu, X.; Peng, H.; Zheng, N.; Yang, Y.; Hu, H.; and Yuan, Y. 2023. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14420–14430.

Lozhkov, A.; Li, R.; Allal, L. B.; Cassano, F.; Lamy-Poirier, J.; Tazi, N.; Tang, A.; Pykhtar, D.; Liu, J.; Wei, Y.; Liu, T.; Tian, M.; Kocetkov, D.; Zucker, A.; Belkada, Y.; Wang, Z.; Liu, Q.; Abulkhanov, D.; Paul, I.; Li, Z.; Li, W.-D.; Risdal,

M.; Li, J.; Zhu, J.; Zhuo, T. Y.; Zheltonozhskii, E.; Dade, N. O. O.; Yu, W.; Krauß, L.; Jain, N.; Su, Y.; He, X.; Dey, M.; Abati, E.; Chai, Y.; Muennighoff, N.; Tang, X.; Oblokulov, M.; Akiki, C.; Marone, M.; Mou, C.; Mishra, M.; Gu, A.; Hui, B.; Dao, T.; Zebaze, A.; Dehaene, O.; Patry, N.; Xu, C.; McAuley, J.; Hu, H.; Scholak, T.; Paquet, S.; Robinson, J.; Anderson, C. J.; Chapados, N.; Patwary, M.; Tajbakhsh, N.; Jernite, Y.; Ferrandis, C. M.; Zhang, L.; Hughes, S.; Wolf, T.; Guha, A.; von Werra, L.; and de Vries, H. 2024. StarCoder 2 and The Stack v2: The Next Generation. arXiv:2402.19173.

McInnes, L.; Healy, J.; and Melville, J. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*.

Olsson, C.; Elhage, N.; Nanda, N.; Joseph, N.; DasSarma, N.; Henighan, T.; Mann, B.; Askell, A.; Bai, Y.; Chen, A.; et al. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Rai, D.; Zhou, Y.; Feng, S.; Saparov, A.; and Yao, Z. 2024. A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models. *arXiv preprint arXiv:2407.02646*.

Räuker, T.; Ho, A.; Casper, S.; and Hadfield-Menell, D. 2023. Toward Transparent AI: A Survey on Interpreting the Inner Structures of Deep Neural Networks. In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 464–483.

Venna, J.; and Kaski, S. 2001. Neighborhood Preservation in Nonlinear Projection Methods: An Experimental Study. In *International Conference on Artificial Neural Networks*.

Vig, J. 2019. A Multiscale Visualization of Attention in the Transformer Model. In Costa-jussà, M. R.; and Alfonseca, E., eds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 37–42. Florence, Italy: Association for Computational Linguistics.

Wang, K. R.; Variengien, A.; Conmy, A.; Shlegeris, B.; and Steinhardt, J. 2023. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 Small. In *The Eleventh International Conference on Learning Representations*.

Yao, T.; Li, Y.; Pan, Y.; and Mei, T. 2024. Hiri-vit: Scaling vision transformer with high resolution inputs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zou, A.; Phan, L.; Chen, S.; Campbell, J.; Guo, P.; Ren, R.; Pan, A.; Yin, X.; Mazeika, M.; Dombrowski, A.-K.; Goel, S.; Li, N.; Byun, M. J.; Wang, Z.; Mallen, A.; Basart, S.; Koyejo, S.; Song, D.; Fredrikson, M.; Kolter, J. Z.; and Hendrycks, D. 2023. Representation Engineering: A Top-Down Approach to AI Transparency. arXiv:2310.01405.