Emergence of symbolic abstraction heads for in-context learning in large language models

Ali Al-Saeedi and Aki Härmä
Department of Advanced Computing Sciences (DACS), Faculty of Science and Engineering, Maastricht University, The Netherlands

Abstract

Large Language Models (LLMs) based on self-attention circuits are able to perform, at inference time, novel reasoning tasks, but the mechanisms inside the models are currently not fully understood. We assume that LLMs are able to generalize abstract patterns from the input and form an internal symbolic internal representation of the content. In this paper, we study this by analyzing the performance of small LLM models trained with sequences of instantiations of abstract sequential symbolic patterns or templates. It is shown that even a model with two layers is able to learn an abstract template and use it to generate correct output representing the pattern. This can be seen as a form of symbolic inference taking place inside the network. In this paper, we call the emergent mechanism abstraction head. Identifying mechanisms of symbolic reasoning in a neural network can help to find new ways to merge symbolic and neural processing.

1 Introduction

Recognizing abstract patterns is a fundamental ability that humans have, allowing them to generalize from a few instances and make inferences on unseen scenarios. LLMs seem to be able to perform similar reasoning tasks and even exceed human performance in some cases (Biever, 2023). Symbolic machine reasoning systems have a long history (Turing, 1950; Berkeley, 1959; Wiener, 1965) but the emergence of the capability in current machine learning systems is not fully understood. Large transformers (Vaswani et al., 2017) and state-space models (Gu and Dao, 2024) exhibit intriguing emergent properties. Extremely large models appear capable of executing tasks like in-context learning (Brown et al., 2020) and chain-of-thought reasoning, which are not directly derivable from their training data. The reasoning abilities of LLMs with tabular, non-text data have been illustrated recently, as noted in (Jiang et al., 2024). Furthermore, the reasoning prowess of LLMs has been highlighted in robotic control (Zeng et al., 2023), autonomous vehicle navigation (Chen et al., 2023), and the processing of IoT sensor data (An et al., 2024).

The *mechanistic interpretability* of large language models (LLMs) remains a highly active field of research, with numerous recent theories about how specific behaviors manifest in such extensive models (Wei et al., 2022; Nichani et al., 2024; Allen-Zhu and Li, 2024; Huang et al., 2023). This line of research is driven by the common understanding that current LLMs are computationally extremely expensive and environmentally unsustainable, for most use cases, and still from their theoretical capacity (Härmä et al., 2024). The current methods for the minimization of the models, e.g., using distillation techniques, produce only relatively small gains (Xu et al., 2024). A better understanding of the mechanisms can help to improve the design of LLM architectures and training paradigms.

The *induction head* mechanism is considered a key factor behind in-context learning, enabling a language model to identify a recurring pattern from its input and either replicate it in the output or merge it with previously stored knowledge (Olsson et al., 2022). Other theories explaining the emergence of certain behaviors in large language models include the concept of task-vectors (Hendel et al., 2023; Akyürek et al., 2023) and Bayesian inference occurring during the model's inference phase (Xie et al., 2022).

In this paper, we investigate the ability of small transformer models to recognize, learn, and generalize abstract sequential symbolic patterns, or templates. A template refers to an abstract sequential symbolic pattern that follows a defined structure but can be instantiated with different symbolic elements. For example, the template $ABCABCAB$ represents a repeated sequence where A, B, and C

are symbolic placeholders that can take on specific values. Templates represent the underlying structure of patterns, allowing us to explore whether models can learn these abstract patterns and generalize to new instances that the model has never seen, such as $ABACABAC$, that follow similar structural rules. These templates can be instantiated into specific sequences,such as 12312312 or 45645645, by assigning values to the placeholders. Understanding whether models can generalize across unseen instantiations and solve such patterns dynamically during inference is in the focus of this work.

Furthermore, we aim to determine whether a model can recognize such instantiations in-context, that is, whether it can recognize the symbolic mappings during inference without retraining and by using this understanding to solve new instances of the same abstract template. This ability would indicate that the model is not only learning patterns from its training data but also reasoning dynamically based on the input it encounters during inference.

Our experiments demonstrate that small transformer models with two or three layers can successfully solve the task of abstract pattern matching, whereas single-layer models fail to solve the abstract task, which aligns with the theory of single-layer models inability to perform the induction head task (Sanford et al., 2024). Additionally, we observed the emergence of an *abstraction head*. We define an *abstraction head* as an attention mechanism in transformer models that attends to previous instantiations of a pattern in an abstract manner, identifying the structural relationships between symbolic placeholders, and using this information to perform pattern matching on unseen instances with a similar abstract structure.

The identification of the mechanisms of symbolic reasoning emerging in the training of neural networks can help to build new types of neuro-symbolic processing paradigms. Moreover, it may also help to train neural networks with an internal visible symbolic reasoning mechanism. The recent survey by Bhuyan et al. (2024) gives a taxonomy of different neurosymbolic systems. The explicit training of abstraction head configurations could be seen as novel way to implement Type 6 neuro-symbolic systems in their taxonomy.

2 Methodology

To achieve our primary objective of understanding how LLMs perform on abstract sequential symbolic patterns, we designed a controlled experimental setup involving synthetic datasets of templates and their instantiations. To generate the data, we define abstract patterns of length eight using three symbolic variables: A, B, and C. For example, consider the abstract pattern $ABCABCAB$. By assigning specific values to A, B, and C, we generate different instantiations of the same template such as 12312312, 45645645, 78978978, and 15915915.

We generated all possible permutations of length 8, resulting in 6561 patterns. During this process, we observed that some patterns did not include all three variables, such as the pattern $ACACACAC$ which only contains variables A and C, leaving out B. To ensure uniformity, we excluded such patterns, requiring that all three variables are present. Additionally, we excluded patterns where the last token appeared only once at the end (e.g. $ABABABAC$), as the last digit in each pattern serves as an evaluation metric in our experimental setup. To avoid duplication, we also treated patterns that are equivalent after instantiation as duplicates. For example, $ABCABCAB$ and $ACBACBAC$ were considered the same pattern and only one was used. This resulted in 1,806 unique patterns, which we split into 80% for training and 20% for testing to evaluate whether the models can learn new abstract patterns they have never encountered before from context.

After generating the abstract patterns, the next step is to concretize them using instantiations. This is achieved by generating all possible unique combinations of variables A, B, and C, ensuring that each variable is different. Per pattern, we obtain 504 different instantiations.

To prepare the data for training and evaluation, we combined every four instantiation in each input sequence, where every instance should represent the same abstract pattern. This choice ensures that the model is exposed to different representations of the same abstract structure, providing sufficient context for generalization. Furthermore, it ensures that the model is exposed to at least one instantiation of a pattern early in the sequence, before generalizing on the rest of the input sequence, which is a crucial pattern in determining the model's ability of learning during inference time. For instance, the abstract pattern $ABCABCAB$ could produce the following input sequence:

[12312312|45645645|78978978|15915915]

To further test the model's ability, we added the unique values of the variables of each instantiation at the end of the input sequence. Since the variables A, B and C are abstract, we added the variables in the order they appeared first, without arranging them according to the abstract pattern. For example, for the pattern $ACCABBAC$, we append ACB at the end, instead of ABC, which reflects what appears first in the instantiation rather than the abstract pattern itself. Finally, the previous input sequence example would be altered to look as follows: [12312312|45645645|78978978|15915915|123|456|789|159]

In a complementary experiment, we tested the model by placing the variables A, B, and C at the beginning of the input sequence. This experiment exposed the model to the variables first before requiring it to perform the pattern matching task. For instance, the input sequence mentioned earlier would be tweaked to be: [123|456|789|159|12312312|45645645|78978978|15915915]

3 Models

To evaluate the ability of small LLM models to generalize and learn abstract patterns, we experiment with several transformer (Vaswani et al., 2017) architectures, designed to test the models capabilities. The primary task for training these models is autoregressive sequence prediction (Brown et al., 2020). In this setup, the model predicts the n_{th} token based on the n-1 previous tokens, using them as context to predict the next output. As for the main experiment, we focus on experimenting with models with one, two, and three layers consisting of eight, four and two attention heads, respectively. Meanwhile, we focus on the 2-layer, 4-head architecture for the second experiment. The hidden size dimension is set to 128 across all architectures, and each model includes a single feed-forward layer. Absolute positional embeddings are used to encode positional information in the input sequences. In this paper, we used the python library X-transformer to build the transformers(Wang, 2024) . As for the training, the models were trained on a batch size of 64 using Adam's optimizer (Kingma and Ba, 2017), publicly available in PyTorch with a learning rate of 1×10^{-3} for a total of 100,000 steps.

4 Evaluation

To assess the ability of the models to generalize and learn new patterns, we designed two evaluation tasks for the main experiment: the last token prediction and variable matching. Accuracy is the primary evaluation metric for both tasks. For the last token prediction task, accuracy measures the proportion of correct predictions for the last digit in the 4th instantiation. For example, given the input:

[12312312|45645645|78978978|1591591**5**]

the model predicts the bolded final token (5). This task evaluates whether the model can correctly recognize the structure of abstract patterns and generalize to the final token over the unseen abstract patterns in the test set. The second task evaluates the model's ability to match variable mappings in the sequence. Given the previous input sequence, the model should predict that the next set of tokens should be: $[1, 2, 3|4, 5, 6|, 7, 8, 9|1, 5, 9]$, and the proportion of those variables predicted correctly over the test set, represents the second task of variable matching. We will also look briefly at the training loss to compare the overall performance of the models.

For the complementary experiment, we focus on measuring the accuracy of the second, third, and fourth instantiations, since the model would have seen both the variables and one instantiation, allowing it to generalize on the rest of the instantiations.

5 Results

In this section, we present the results of training the models on instantiations based on abstract patterns, and testing them on the instantiations of the abstract patterns in the test set, which the model has not been exposed to. We will include the training loss of the models, in addition to the last token prediction and variable matching metrics. In addition, we visualize specific attention heads based on their importance and contribution to solving the tasks.

5.1 3-Layers&2-Head

Figure 1 shows the training loss for different runs of the 3-Layer, 2-head model. Although all models converge to the same training loss(0.68), different runs exhibit different behaviors. For instance, we notice bumps emerging in the training loss. Specifically, the model represented in green experiences a bump that occurs after approximately 15,000 steps, and the model represented in pink, where a similar

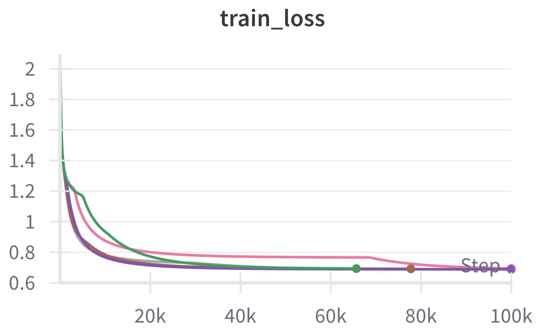

Figure 1: Training loss of five different runs of the 3-Layer, 2-Head model

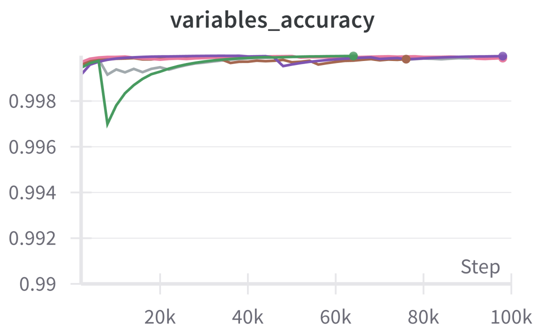

Figure 2: Accuracy of the variable matching task of five different runs of the 3-Layer, 2-Head model

bump occurs after approximately 70,000 steps. We can also see a correspondence between the number of steps where the bumps occur, and the number of steps before the sudden rise in the last digit accuracy shown in Figure 3 for both models represented in green and pink. Figure 3 also shows that not all models achieve perfect accuracy, with two models reaching an accuracy of approximately 0.98. Meanwhile, Figure 2 shows that all models successfully complete the variable matching task, achieving an accuracy of approximately 1.0 across all runs.

To further evaluate the performance of the 3-layer, 2-head model, we showcase an example from the test set, highlighting how one of the runs(represented in gray) solves the pattern.

Predicted pattern by the 3l2h Model:

[? 3 3 1 3 3 1 3 | 7 6 4 2 5 7 6 7 | 4 9 7 5 6 8 9 8 | 7 9 6 3 1 5 9 5 | 3 1 7 | 6 2 7 | 9 5 8 | 9 3 5]

Correct pattern:

[3 3 1 1 7 7 3 7 | 6 6 2 2 7 7 6 7 | 9 9 5 5 8 8 9 8 | 9 9 3 3 5 5 9 5 | 3 1 7 | 6 2 7 | 9 5 8 | 9 3 5]

To understand how the model solves the prob-

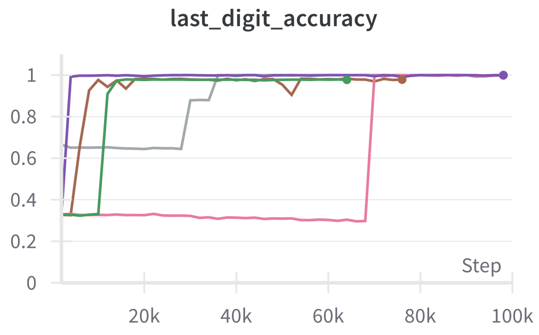

Figure 3: Accuracy of the last digit task of five different runs of the 3-Layer, 2-Head model

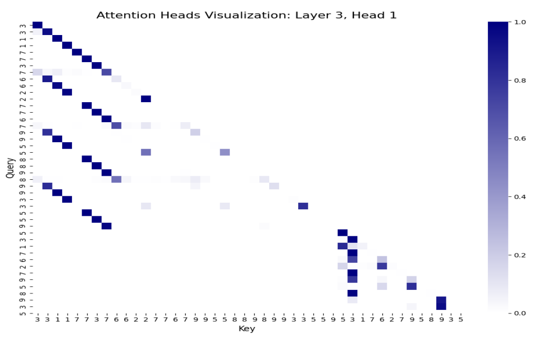

Figure 4: Attention head responsible for abstraction in 3-Layer, 2-Head model

lem, we visualized the attention patterns of solving a test instance during inference time. Among the six attention heads, two attention heads provided us with useful insight into how both tasks are solved(Figures 4 and 5). Figure 4 visualizes the abstraction of patterns, where the attention mechanism focuses on the relationship between the first instantiation and the last three instantiations. Specifically, the tokens in the last three instantiations, which are attending back to the first instantiation to predict the next token. For example, in the second instantiation the bolded 6 in **6**6227767 is attending to the bolded 3 in 3**3**117737 in the first instantiation. This behavior is consistent across all three later instantiations(second, third and fourth). Moreover, this pattern of attention is not only specific to the first token, with nearly all tokens in the instantiations attending back to their corresponding "abstract" next token in the first instantiation, with the exception of the fifth position.

5.2 2-Layers&4-Head

Figure 6 shows the training loss across multiple runs of the 2-layer, 4-head model, where we observe that only one run fails to converge to the

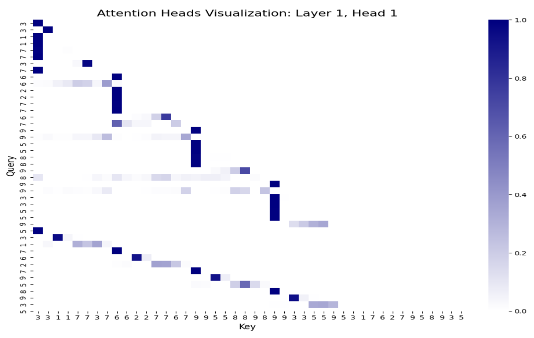

Figure 5: Attention head responsible for variable matching task in 3-Layer, 2-Head model

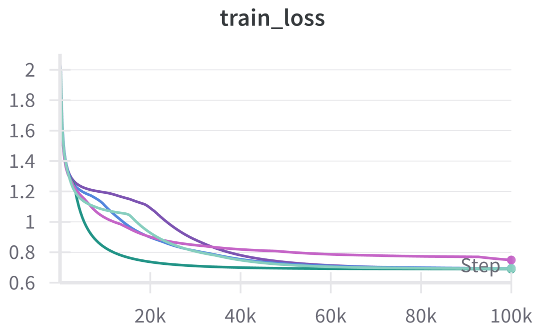

Figure 6: Training loss of five different runs of the 2-Layer, 4-Head model

minimum loss (0.68). Figure 7 shows that the model consistently succeeds in the task of identifying unique variables. Meanwhile, this is not the case for the last-digit prediction task. Only two runs achieved perfect accuracy, two runs achieved near-perfect accuracy between 0.96 and 0.98, and one run reached a maximum accuracy of 0.6.

We now present an example showing how the model predicts an input sequence using one of the best-performing runs (represented in purple). **Predicted pattern by 2l4h model:**

[? 3 3 3 1 1 3 3 | 9 6 1 2 9 7 6 7 | 9 9 1 5 8 8 9 8 | 1 9 3 3 8 5 9 5 | 3 1 7 | 6 2 7 | 9 5 8 | 9 3 5]

Correct pattern:

[3 3 1 1 7 7 3 7 | 6 6 2 2 7 7 6 7 | 9 9 5 5 8 8 9 8 | 9 9 3 3 5 5 9 5 | 3 1 7 | 6 2 7 | 9 5 8 | 9 3 5]

To better understand how the model predicts patterns, we visualize the attention mechanisms of the two most significant heads out of the eight available attention heads in Figures 9 and 10. In Figure 9, most of the tokens in the second, third, and fourth instantiations are attending back to the first instantiation to predict the next token. For instance, we observe the attention of the highlighted

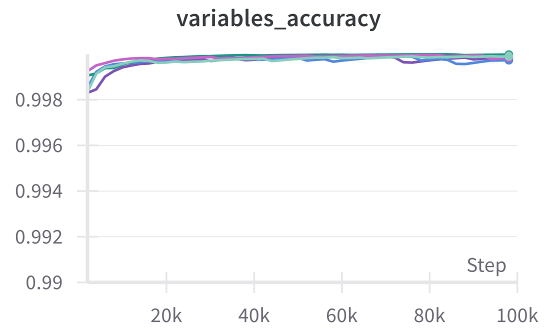

Figure 7: Accuracy of the variable matching task of five different runs of the 2-Layer, 4-Head model

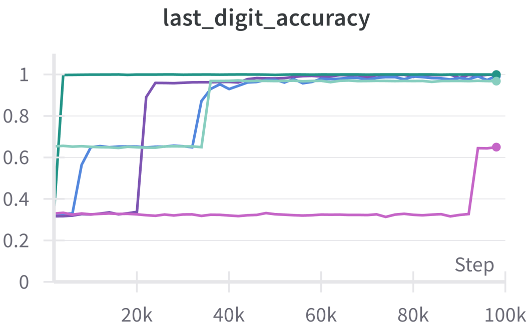

Figure 8: Accuracy of the last digit task of five different runs of the 2-Layer, 4-Head model

digits in the second instantiation, 66227767 , to the corresponding highlighted digits in the first instantiation(excluding black), 33117737. We also observe that the attention is not always directed to the first pattern; at times, it shifts between other instances, such as the bolded 8 in the third instantiation: 9955**8**898 attending back to the bolded 7 in the second instantiation: 66227**7**67, which is abstractly the next token. Figure 10 shows the attention mechanism used to solve the second task of matching the variables, where we observe that the attendance was on the correct next token, eleven out of twelve times.

5.3 1-Layer&8-Head

Figure 11 shows the training loss across four runs of the 1-layer, 8-head model, where all the runs fail to converge to the minimum training loss. This failure is reflected in Figure 13, which shows that the model is unable to solve the last digit prediction task. Thus, we do not present any examples of the model's predictions. In contrast, Figure 12 shows that all runs successfully performed the variable

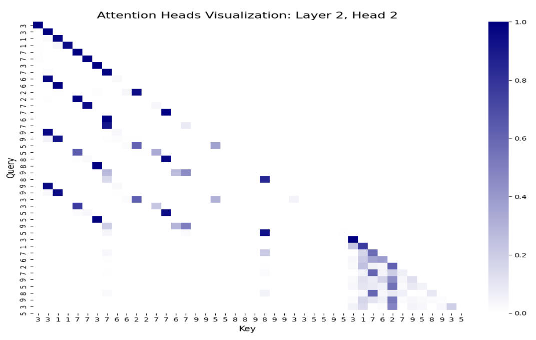

Figure 9: Attention head responsible for abstraction in 2-Layer,4-Head model

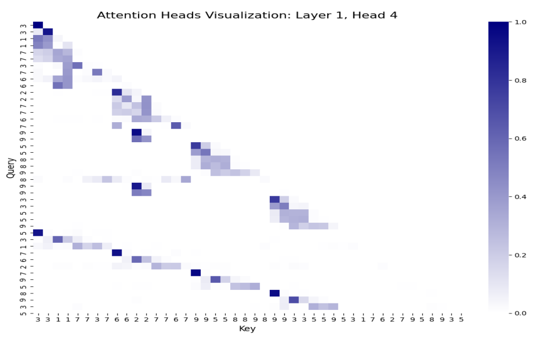

Figure 10: Attention head responsible for the variable matching task in 2-Layer, 4-Head model

matching task.

5.4 2-Layer&4-Head(Experiment 2)

Another experiment involved appending the variables at the beginning of the sequence, followed by the instantiations of the pattern. As shown in Figure 15, all models eventually converge to perfect accuracy. Figure 16 shows the attention head responsible for abstraction in this task, where the second, third, and fourth instantiations attend back to the first instantiation.

6 Discussion

This section analyzes the results presented in the previous section, focusing on the model's performance on the training loss and the accuracy metrics we defined, attention mechanisms, and the models' ability to generalize on abstract patterns during inference time.

6.1 3-Layer&2-Head

Referring back to Figure 1, we previously noted that all models converged to a training loss of approximately 0.68 which we assume to be the lowest achievable loss for this task. This limitation is caused by the model's lack of knowledge about the abstract pattern and the specific values of the variables A ,B and C which it only learns during inference, relying on a trial-and-error process.

We also observed bumps in the training loss that correspond to the sudden rise in accuracy shown in Figure 3. We hypothesize that this reflects a change in the models' behavior, where the models learns a critical strategy that allows it to generalize over the instances in the training set, thus improving its performance on the test set. In Figure 3, we observed that two out of five models did not achieve perfect accuracy but still managed to achieve a minimum accuracy of 0.98. Although we did not find a justification on why it is not acting like the other three models, we can still conclude that the 3-layer, 2-head model succeeds in this task, as all runs can generalize and achieve near-perfect accuracy.

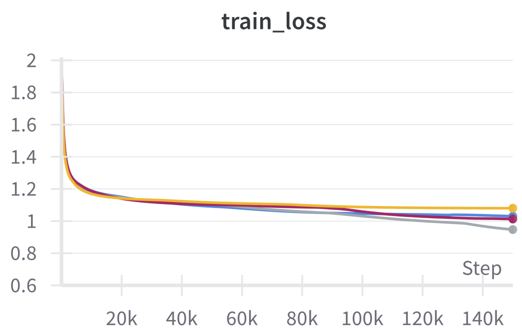

Figure 11: Training loss of four runs of the 1-Layer, 8-Heads model

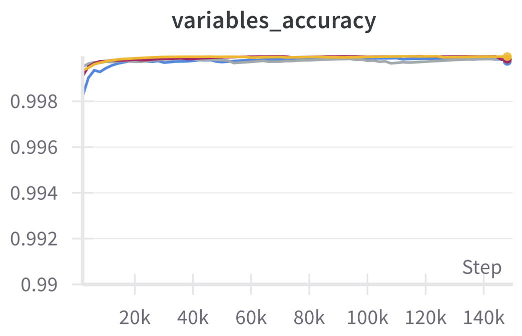

Figure 12: Accuracy of the variable matching task of five different runs of the 1-Layer, 8-Head model

6.2 The abstraction head

We now proceed to analyze the model's predictions(see 5.1) across the four instantiations to better

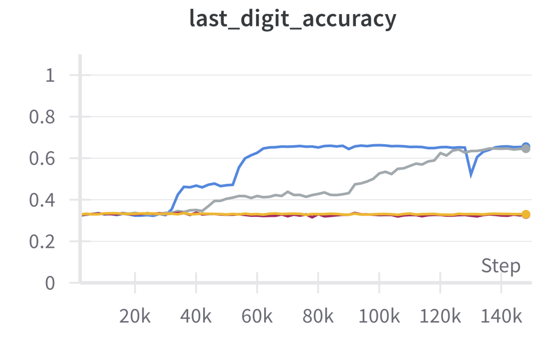

Figure 13: Accuracy of the last digit in the fourth instance of a pattern of four runs of the 1-Layers 8 heads model

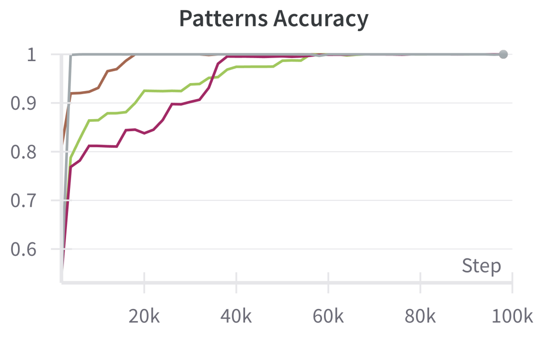

Figure 15: Accuracy of the last three instantiations over the test set

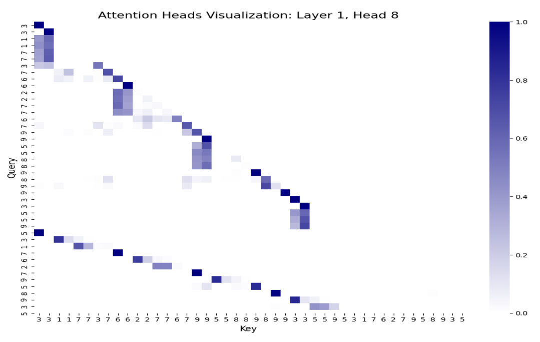

Figure 14: Attention head responsible for variable matching task in 1-Layer, 8-Head model

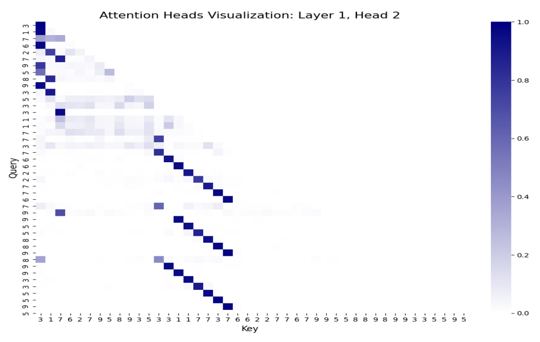

Figure 16: Attention head responsible for abstracting in 2-Layer, 4-Head model

understand its reasoning process and generalization capabilities. The model begins by predicting random tokens in the first eights positions due to the lack of prior information on the abstract pattern and the variables. After predicting the first eight tokens and receiving the prior context, the model gains two pieces of information: the abstract pattern is $AABBCCAC$ and the first three unique digits are $A = 3$, $B = 1$ and $C = 7$. Using this feedback, the model proceeds to predict the next instance. Initially, it tried to predict that the value of A is nine, but updates its understanding after receiving feedback on the correct value of A. The model then correctly predicts the value of A in the following position based on the feedback received, and predicts six. This process is repeated for the next four tokens, $BBCC$, where the model similarly predicts the unseen values of B and C, refining its predictions based on the feedback received. For the last two positions: 662277**67**, the model uses its prior knowledge that the abstract pattern is $AABBCCAC$ and that the values of the variables in the second instance are $A = 6$, $B = 2$, and $C = 7$ to predict the last two values correctly. The same logic is applied to the third and fourth instantiation. Figure 4 shows the attention mechanism we assume is mainly responsible for solving the task of abstract pattern matching. In Figure 4, we observe that the current position in the second, third, and fourth instantiations consistently attends to the corresponding next position in the first instantiation. For example, we mentioned previously, the bolded 6 in **6**6227767 from the second instantiation attends to the bolded 3 in 3**3**117737 from the first instantiation. This can be seen as the bolded A in **A**$ABBCCAC$ attending to the next position, represented as the bolded A in A**A**$BBCCAC$. We identify this attention head as the one responsible for abstracting the patterns, and we refer to it as the *Abstraction Head*, AH. This head seems to have developed the ability to look back at first eight tokens, or the first instantiation and attend to its abstract form. This head aligns with the concept of induction heads, particularly in its ability to perform pattern completion, such as $[A^*][B^*] \dots [A] \rightarrow [B]$, where $A^* \approx A$ and $B^* \approx B$ (Olsson et al., 2022). However, the abstraction head we found operates at a more abstract level, focusing on pattern matching according to a template that has not been specified to the model, and the model was able to figure out from only instantiations of the data about this abstract pattern. In Figure 2, we observe that the models succeed in the variable matching task within the first 2,000 steps. This task appears to require no abstraction, as the models develop a straightforward matching strategy, which is also shown in the attention pattern shown in Figure 5.

6.3 2-Layer&4-Head

Regarding the training loss of multiple runs of the 2-layer, 4-head architecture, as shown in Figure 6, we observe that four out of five runs successfully converge to the minimum loss. We hypothesize that the failed run is a result of undertraining and with enough training, all models are able to reach a near-perfect accuracy. We also observe a correlation between the last digit accuracy, as shown in Figure 8, and the training loss, specifically, the run represented in pink, which failed to converge to the minimum training loss, and was unable to achieve perfect accuracy in the last digit prediction task. Regarding the model's prediction (see 5.2), we observe that it uses a similar approach to the 3-layer, 2-head model. Specifically, the model relies on a trial-and-error process when the values of the variables were unknown and started to guess random digits when it did not have any pieces of information about the abstract pattern. We also see that one of the attention mechanisms in the attention heads (Figure 9) is similar to the mechanism observed in Figure 4, specifically what we refer to as an *abstraction head*. However, the abstraction head in Figure 9 not only attends to the first instantiation but also sometimes focuses on the previous instantiations (second and third). While what causes this behavior remains unclear, the model still uses a valid approach to solve the pattern matching task. As shown in Figure 7, all runs successfully performed the variable matching task. Furthermore, Figure 10 reveals that the attention mechanism used for this task is very similar to the one observed in Figure 5, suggesting that the variable matching task is straightforward, where the model develops an attention head that performs a simple tracking back to the digits and predicts the correct variables.

6.4 1-Layer&8-Head

For this architecture, we observed that 1-layer might not be sufficient to solve the last digit prediction task, as shown in Figure 13. However, we found that a single layer can still be used to tackle non-abstract tasks, such as variable matching, which does not require abstraction. This is demonstrated in Figure 12. The mechanism used for this task, shown in Figure 14, aligns with the behavior observed in other models.

6.5 2-Layer&4Heads(Experiment 2)

The complementary experiment confirmed that 2-layer models are capable of solving the task of abstract pattern matching, as demonstrated in Figure 15. Additionally, we observed the emergence of the *abstraction head*, illustrated in Figure 16. However, unlike the previous task, we found that some models were able to solve this task without developing similar abstraction heads. This suggests that these models may have discovered an alternative mechanism to solve the problem, which needs further investigation in future work.

7 Conclusions

In this study, we investigated the ability of small transformer models to recognize, learn, and generalize abstract sequential symbolic patterns through controlled experiments. We demonstrated that models with two or three layers successfully perform this task, unlike single-layer models. A key finding was the emergence of the *abstraction head*, an attention mechanism that directs its focus toward the first instantiation in an abstract manner. Our findings on the emergence of abstraction heads provide a foundation for advancing neuro-symbolic processing paradigms, potentially enabling the development of new Type 6 neuro-symbolic systems which contain symbolic processing inside a trained neural network.

8 Limitations of the work

The results of this paper are based on a very simplified experiments based on sequences of numbers. It is possible that the proposed abstraction mechanism is not a primary mechanism in reasoning tasks based on human language or in cases where the entities have more complex relations. The study was also limited to small transformer models with only 1-3 layers of self-attention models. It is possible that different abstraction mechanisms emerge in models of more layers, which may not be visible in the experiments reported in this paper.

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2023. What learning algorithm is in-context learning? Investigations with linear models. *arXiv preprint*. ArXiv:2211.15661 [cs].

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. *arXiv preprint*. ArXiv:2309.14316 [cs].

Tuo An, Yunjiao Zhou, Han Zou, and Jianfei Yang. 2024. IoT-LLM: Enhancing Real-World IoT Task Reasoning with Large Language Models. *arXiv preprint*. Version Number: 2.

Edmund Callis Berkeley. 1959. *Symbolic Logic and Intelligent Machines*. Literary Licensing, LLC.

Bikram Pratim Bhuyan, Amar Ramdane-Cherif, Ravi Tomar, and T. P. Singh. 2024. Neuro-symbolic artificial intelligence: a survey. *Neural Computing and Applications*, 36(21):12809–12844.

Celeste Biever. 2023. ChatGPT broke the Turing test — the race is on for new ways to assess AI. *Nature*, 619(7971):686–689. Bandiera_abtest: a Cg_type: News Feature Publisher: Nature Publishing Group Subject_term: Computer science, Mathematics and computing, Technology, Society.

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

Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. 2023. Driving with LLMs: Fusing Object-Level Vector Modality for Explainable Autonomous Driving. *arXiv preprint*. ArXiv:2310.01957 [cs].

Albert Gu and Tri Dao. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint*. ArXiv:2312.00752 [cs].

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-Context Learning Creates Task Vectors. *arXiv preprint*. ArXiv:2310.15916 [cs].

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv preprint*. ArXiv:2311.05232 [cs].

Aki Härmä, Marcin Pietrasik, and Anna Wilbik. 2024. Empirical Capacity Model for Self-Attention Neural Networks. *arXiv preprint*. ArXiv:2407.15425 [cs, stat].

Ruya Jiang, Chun Wang, and Weihong Deng. 2024. Seek and Solve Reasoning for Table Question Answering. *arXiv preprint*. ArXiv:2409.05286 [cs].

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Eshaan Nichani, Alex Damian, and Jason D. Lee. 2024. How Transformers Learn Causal Structure with Gradient Descent. *arXiv preprint*. ArXiv:2402.14735 [cs, math, stat].

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. *Preprint*, arXiv:2209.11895.

Clayton Sanford, Daniel Hsu, and Matus Telgarsky. 2024. One-layer transformers fail to solve the induction heads task. *Preprint*, arXiv:2408.14332.

A. M. Turing. 1950. Computing machinery and intelligence. *Mind*, LIX(236):433–460.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv preprint*. ArXiv:1706.03762 [cs].

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. *Preprint*, arXiv:2305.14160.

Phil Wang. 2024. lucidrains/x-transformers. Original-date: 2020-10-24T22:13:25Z.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. *arXiv preprint*. ArXiv:2206.07682 [cs].

Norbert Wiener. 1965. *Cybernetics, Second Edition: or the Control and Communication in the Animal and the Machine*, 2nd edition edition. Mit Pr, Cambridge, MA, USA.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An Explanation of In-context Learning as Implicit Bayesian Inference. *arXiv preprint*. ArXiv:2111.02080 [cs].

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024. A Survey on Knowledge Distillation of Large Language Models. *arXiv preprint*. ArXiv:2402.13116.

Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S. Yu. 2023. Large Language Models for Robotics: A Survey. *arXiv preprint*. ArXiv:2311.07226 [cs].

A Appendix: Abstraction Heads

One aspect we explored was whether the abstraction head tends to emerge in a specific layer(e.g. the first or the last). Figure 17 shows the visualization of another successful run of the 3-Layers, 2-Head architecture. The abstraction head shown in Figure 4 emerges in the third layer, while the abstraction head in Figure 17, emerges in the second layer. However, even tho attention heads tend to appear in the last few layers, none of the experiments showed an abstraction head in the first layer. Research showed that attention heads that extract and makes use of critical information such as label appear in deep layers (Wang et al., 2023). This might suggest that abstraction heads appear in Deep Layers(i.e. layers closer to the output), and might show that attention heads that require abstraction(label, abstract pattern, etc.), are usually closer to the output, even in small models, which is a matter of further investigations.

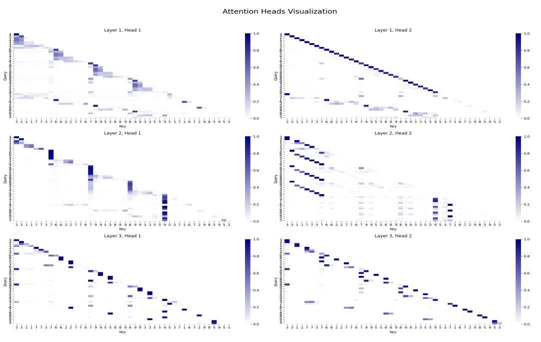

Figure 17: Visualization of all attention heads from another run