Extended Abstract

Motivation This project was motivated by the introduction of Chain of Thought reasoning models and its affect on the growing computational cost of interacting with large language models (LLMs) due to increasingly long input prompts. While much work has gone into compressing models themselves, we explore the less-studied but promising direction of compressing inputs. This allows the model to not sacrifice on performance computationally. We aim to only keep information that is important to the model, yielding efficiency gains and interpretability ones as well. By learning what is important for a model to learn, it could give insight for humans as to what is important for them to learn.

Method We introduce Compression of Thought, a reinforcement learning framework that trains language models to compress prompts using purely unsupervised reward signals. Our approach formulates prompt compression as an RL problem where a trainable model learns a policy to maximize a reward that balances task accuracy and compression ratio. The core technical innovation lies in our reward function design and training procedure. Unlike existing methods that rely on semantic similarity metrics or supervised compression targets, we optimize directly on downstream task performance. Our reward function provides the only learning signal and only uses compression ratios and task performance, where accuracy rewards are computed by evaluating a frozen model's performance on compressed prompts, and compression rewards incentivize shorter outputs.

We employ Group Relative Policy Optimization (GRPO) rather than standard policy gradient methods, enabling stable training with our multi-objective reward function. The model is trained using the MultiRC dataset, a subset of the SuperGlue dataset.

Implementation We use a subset of the MultiRC dataset due to memory constraints, where each sample consists of a passage, a question, and a possible answer, and a binary yes/no answer. The trainable model receives the passage and generates a compressed prompt. Then, an evaluation prompt is created, instructing the model to determine whether the possible answer is correct. The response to this, in addition to the compression ratio, is used to calculate a reward. The reward function is calculated without using semantic similarity, optimizing purely on performance on the dataset. Training is done using a GRPO implementation tailored to this setup, using OLMo and LLaMA models in various frozen/trainable model pairings.

Results Quantitatively, LLaMA-based models converged faster and produced higher reward scores compared to OLMo-based models. On average, LLaMA compressions were 88.15 tokens shorter and yielded a 0.05898 improvement in reward. Qualitatively, LLaMA-generated compressions were often structured as factual lists, while OLMo tended to anticipate questions, showing divergent strategies despite identical reward signals and datasets. Examples show successful compressions preserving key information while drastically reducing prompt length.

Discussion The divergent behaviors between model architectures suggest that compression strategies are deeply influenced by model priors and not just the reward signal. Despite the lack of instruction, models learned to preserve question-relevant content. The small dataset size (1,520 examples) and limited compute constrained the scope of experiments. Implementation challenges (including PPO instability) led to switching to GRPO. Nonetheless, the method demonstrates that reinforcement learning can guide models to discover task-relevant compression policies autonomously.

Conclusion *Compression of Thought* presents a novel, unsupervised approach to prompt compression, offering benefits for both efficiency and interpretability in LLM use. While current results are preliminary, they suggest that models can learn to compress input while preserving task performance, without explicit guidance. Future work will expand to larger datasets, longer training runs, and stronger frozen models to better test the limits and capabilities of learned compression strategies.

Compression of Thought

Andrew Lanpouthakoun
Department of Computer Science
Stanford University
andlanpo@stanford.edu

Abstract

We explore novel techniques in reinforcement learning and prompt compression. To interact with a large language model, one must prompt it. As these messages become increasingly large (with quantity of tokens being used increasing monotonically), calls become extremely computationally expensive. With many approaches to quantizing models targeting the size of the model and pruning, we hope to target the inputs instead. We hope to boost the performance of specifically Chain-of-Thought Reasoning Models by decreasing the quantity of input tokens, leading to a naturally more quantized model. Our approach not only yields results in the domain of optimization, but also leads to results in the field of interpretability due to its uninstructed summarization technique.

1 Introduction

Chain-of-Thought (CoT) reasoning models have shown to be excellent at answering complicated mathematical problems and reasoning through logic. However, many of these models iterate through too much information and "overthink". This overthinking issue (Chen et al. 2025)[1] adds to the expensive computation that non-Chain-of-Thought models experience already. To make this more efficient, it would be useful for models to distill the most crucial information from a prompt before "overthinking".

Even without looking at CoT models, typical Large Language Models already face issues dealing with large contexts. As context grows in length, models begin to hallucinate more often, with many models typically having a fixed context window. To converse over a long period of time with a LLM requires extremely high levels of computational power, often leading to entire truncation and making it difficult for users without powerful compute to run models locally.

This task of summarization bring in the issue of interpretability of Large Language Models that continues to evade researchers. The recent release of Mistral's Sliding Window Attention technique (Jiang et al. 2023) [5] has shown that attention to every word in a sequence is not completely computationally necessary. Compression of Thought hopes to target both of these ideas, find what is truely important to models, and what can we glean from this information. Can we train a model to only retain information that it truly finds interesting? What can we learn from that?

This research focuses on improving both of these tasks, Compression of Thought hopes to learn to shrink queries with exclusively an unsupervised reward signal. Our outputs are a summarized version of the input text. By doing this with small language models (Groeneveld et al. 2024)[4]) the amount of computation required to complete full queries can go down exponentially, only by pre-appending or appending a comparably tiny model.

2 Related Work

2.1 Compression Methods

2.1.1 LLM-Lingua 2

LLM-Lingua - 2 (Pan et al. 2024).[7] is a frame work that compresses prompts using supervised learning methods. It first creates a dataset using GPT-4, instructing the model to compress texts by only discarding words that it deems unnecessary. Then, it trains a Transformer model to learn to delete or preserve tokens independently. While this does seem to work well at compressing, it doesn't generalize well on tasks outside of the dataset because it learns exclusively on certain tasks. Compression of Thought hopes to use its unsupervised signal to be more general.

2.1.2 TACO-RL

TACO-RL (Shandilya et al. 2024). [9] is a framework that builds upon LLM-Lingua 2 by introducing online RL methods to further finetune on downstream tasks. While LLMLingua 2 is a purely supervised learning task, TACO-RL uses the Vanilla Policy Method to compare the output of the original prompt to the newly summarized prompt. The paper utilizes F1 and BLEU score to compare results. While this seems to be incredibly effective on summarizing based off different tasks, the metrics used are somewhat rudimentary and doesn't allow the model to learn to be better than the original. By comparing outputs of the summarized to the original purely based off similarity, we don't allow the possibility of the new summarized output being better than the original. This is possible because there could be simply less noise in the summarized prompt than in the original. Compression of Thought hopes to improve on this by rewarding based off task performance, by not comparing with the original text, it allows the new model to improve over the original model.

2.1.3 Nano-Capsulator

Nano-Capsulator (Chuang et al. 2024). [3] is a framework which compresses prompts into capsule prompts formatted in natural language. This paper is the most similar to the task we are trying to optimize. It notes that generating texts is an inherently non-differentiable process for which backpropagation does not apply, and instead seeks to optimize a reward function defined by utilizing a reward function which reflects the degree of semantic preservation required for the prompt. This paper also approaches the problem of prompt compression from an unsupervised lens. While this paper does many interesting methods, it does this through some level of instruction by telling a model to summarize explicitly. Our model will have this be purely learned, hoping for our model to learn without instruction. This uninstructed approach will allow a glimpse into what the model values in terms of information.

2.2 Policy Gradient Methods

Policy optimization methods are a large part of modern reinforcement learning, with the goal of learning policies that maximize expected cumulative rewards. Classical approaches like REINFORCE (Williams et al. 1992).[11] introduced the policy gradient theorem, however, this method suffered from high variance and inefficient data usage and since its introduction in the 1990s, several policy gradient advancements have been made.

2.2.1 Proximal Policy Optimization

Proximal Policy Optimization: PPO (Schulman et al. 2017)[8] addresses some stability issues through clipped objectives and trust regions. Our initial experiments with PPO failed due to gradient explosion caused by our multi-objective reward function, highlighting the need for more sophisticated optimization approaches.

2.2.2 Group Relative Policy Optimization

The most recent advancement in Policy Gradient methods is Group Relative Policy Optimization (GRPO) (Shao et al. 2024). [10]. GRPO represents a novel advancement in preference-based policy optimization that addresses key limitations of existing methods. Unlike other approaches

that rely on pairwise comparisons or absolute reward signals, GRPO leverages group-wise relative rankings. GRPO is ideal for our task because it can naturally handle the trade-offs between multiple objectives, due to its relative comparisons. Rather than optimizing absolute values, GRPO can learn to rank different compression strategies based on their relative performance across both dimensions simultaneously.

3 Method

Here, we introduce the methodology of the Compression of Thought algorithm. Compression of Thought closely follows the algorithm outlined in Nano-Capsulator. We train by first inputting a passage from the MultiRC dataset (Khashabi et al. 2018) [6] into our trainable model. Then, we create a prompt including the associated question from the dataset and ask the frozen model to answer the question. The goal is to evaluate whether the amount of information provided by the compressed prompt is enough to answer the question given. We introduce three novel contributions off this method: a lack of supervision, a different reward function, and a streamlined training loop.

3.1 Lack of Supervision

Nano-Capsulator utilizes a compression prompt that is prepended to the original passage encouraging summarization. To remove supervision and in an attempt to analyze what the model values deeply to answer questions, this supervision was removed without replacement. We left the model to learn exclusively from the reward signal.

3.2 Reward Function

Compression of Thought utilizes the following reward function:

- p_i be the original prompt
- c_i be the compressed completion
- q_i be the question
- a_i be the proposed answer
- $l_i \in \{\text{yes}, \text{no}\}$ be the true label
- $\hat{l}_i \in \{\text{yes}, \text{no}, \text{invalid}\}$ be the predicted label

$$\text{original\_length} = \text{len}(p_i), \text{compressed\_length} = \text{len}(c_i)$$

$$r_{\text{comp}}^{(i)} = \begin{cases} 1.0 & \text{if original\_length} = 0 \\ 1 - \text{clip}\left(\dfrac{\text{compressed\_length}}{\text{original\_length}},\ 0.1,\ 3.0\right) & \text{otherwise} \end{cases}$$

$$r_{\text{acc}}^{(i)} = \begin{cases} 1.0 & \text{if } \hat{l}_i = l_i \\ -0.2 & \text{if } \hat{l}_i \in \{\text{yes}, \text{no}\} \wedge \hat{l}_i \neq l_i \\ -0.5 & \text{if } \hat{l}_i = \text{invalid} \end{cases}$$

$$r^{(i)} = 0.6 \cdot r_{\text{acc}}^{(i)} + 0.4 \cdot r_{\text{comp}}^{(i)}$$

This reward function rewards both accuracy and compression, weighting accuracy slightly higher to still encourage sensical compressions. The choice of weights was arbitrary; due to limited time, further weights were not tested. With more time allowed, further weights would be tested.

3.3 Streamlined Training Process

Clearly from the reward function, Compression of Thought does not compare outputs from the original model on the input texts to maintain semantic similarity. In fact, there is no effort to maintain semantic similarity. The reward signal encourages compression that maintains accuracy on questions; the choice of this streamlining was because it seemed more closely aligned with the evaluation metric.

The model hopes to perform well on this dataset given compressed prompts, comparing semantic similarity seemed to give it too many parameters to attempt to optimize at once. Additionally, this is done to make the model run more efficiently, there are now two total passes in a forward step during training rather than three.

Algorithm 1 Compression of Thought Training Algorithm

Require: Dataset $\mathcal{D}$ (MultiRC), trainable model M_θ, frozen model M_f, batch size B
Ensure: Optimized policy parameters θ^*

Initialize trainable model M_θ and frozen model M_f
for each training batch $\mathcal{B} = \{(\text{prompt}_i, \text{question}_i, \text{answer}_i, \text{label}_i)\}_{i=1}^{B}$ **do**
    // Generation Phase
    for each sample $i \in \mathcal{B}$ **do**
        $\text{completion}_i \leftarrow M_\theta.\text{generate}(\text{inputs}_i, \text{max\_new\_tokens} = 256)$
        $\text{original\_length}_i \leftarrow |\text{prompt}_i.\text{split}()|$
        $\text{compressed\_length}_i \leftarrow |\text{completion}_i.\text{split}()|$
    end for
    // Reward Computation Phase
    for each sample $i \in \mathcal{B}$ **do**
        $\text{ratio}_i \leftarrow \text{clip}\left(\frac{\text{compressed\_length}_i}{\text{original\_length}_i}, 0.1, 3.0\right)$
        $\text{compression\_score}_i \leftarrow 1.0 - \text{ratio}_i$
        $\text{eval\_prompt}_i \leftarrow$ `"Based on the following passage, answer whether"`
            `the given answer is correct. Begin Passage: {completion_i}`
            `End Of Passage. Question: {question_i} Proposed Answer:`
            `{answer_i} Is this answer correct? Respond with ONLY`
            `the word 'yes' or 'no'. Answer:"`
        $\text{eval\_prompts.append}(\text{eval\_prompt}_i)$
        $\text{true\_labels.append}(\text{"yes" if label}_i = 1 \text{ else "no"})$
    end for
    // Frozen Model Evaluation torch.no_grad():
    $\text{eval\_inputs} \leftarrow \text{frozen\_tokenizer}(\text{eval\_prompts}, \text{padding} = \text{True})$
    $\text{outputs} \leftarrow M_f.\text{generate}$
    for each sample $i \in \mathcal{B}$ **do**
        $\text{response}_i \leftarrow \text{frozen\_tokenizer.decode}(\text{outputs}[i])$
        $\text{reward}_i \leftarrow 0.6 \times \text{accuracy\_reward}_i + 0.4 \times \text{compression\_score}_i$
        $\text{reward}_i \leftarrow \text{reward}_i + \mathcal{N}(0, 0.01)$ ▷ Add noise for exploration
    end for
    // GRPO Update
    $\text{grpo\_trainer.step}(\text{prompts}, \text{completions}, \text{rewards})$
    if step $\bmod$ logging_steps $= 0$ **then**
        Log training metrics to tensorboard
    end if
    if step $\bmod$ save_steps $= 0$ **then**
        Save model checkpoint
    end if
end for

4 Experimental Setup

4.1 Dataset

We conduct experiments on a reduced subset of the MultiRC dataset. The original dataset consists of 27,243 samples, but for efficiency and feasibility of rapid iteration, we use a randomly selected subset of 1,520 samples. Each sample in MultiRC consists of a passage, an associated question, and multiple candidate answers, each labeled independently as either correct or incorrect. In our experiments, we simplify the format to one question-answer pair per sample and retain only binary answer labels: yes or no.

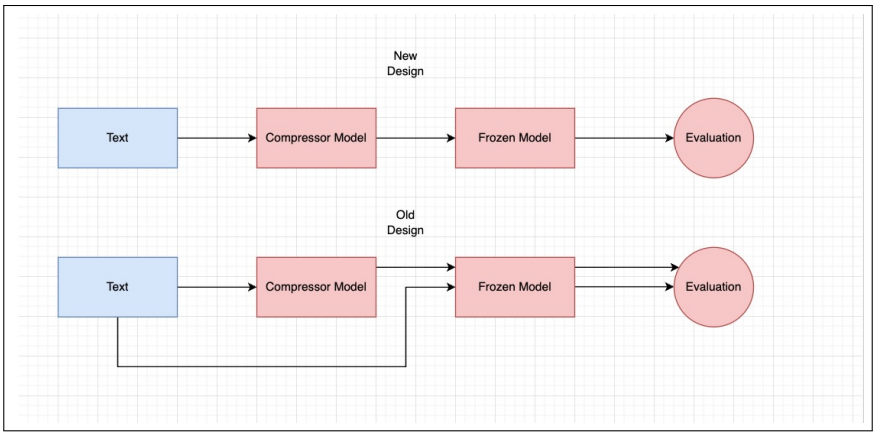

Figure 1: Streamlined vs Original Training Process

4.2 Model

Two models are used in the training process:

- Trainable model: This model receives the full passage and is responsible for producing a compressed prompt. Here we use OLMo-2-1B or LLama 3.2-1B.
- Frozen model: This model remains unchanged throughout training. It is tasked with answering a question based on the compressed prompt generated by the trainable model. Here we also use OLMo-2-1B or LLama 3.2-1B.

Both models are variants of open-source transformer-based language models. Across experiments, we test multiple mixtures of trainable and frozen model pairs to evaluate their effect on compression quality and question-answering performance.

4.3 Training

Compression of Thought is trained using reinforcement learning where each training step involves:

- Feeding the full passage to the trainable model to generate a compressed prompt.
- Constructing a new prompt containing the compressed passage and question.
- Using the frozen model to answer the question based on the compressed prompt.
- Computing a reward based on both the correctness of the answer and the degree of compression (as described in Section 2).

Training is done using GRPO with reward feedback at each step. To streamline computation, each forward pass involves only two model calls.

5 Results

We evaluate Compression of Thought across four model configurations, analyzing both compression efficiency and task performance. Our results demonstrate significant architectural differences in compression learning capabilities and reveal unexpected patterns in how different models approach the compression-accuracy trade-off.

5.1 Quantitative Results

We found that experimental groups that involved LLama as the trainable model performed generally better than groups with OLMo as the trainable model, with an average difference in maximum completion lengths of 88.15 tokens, as seen in Figure 2. The effect of this can be seen in the evaluation rewards, with rewards having an average difference of 0.05898 between base models.

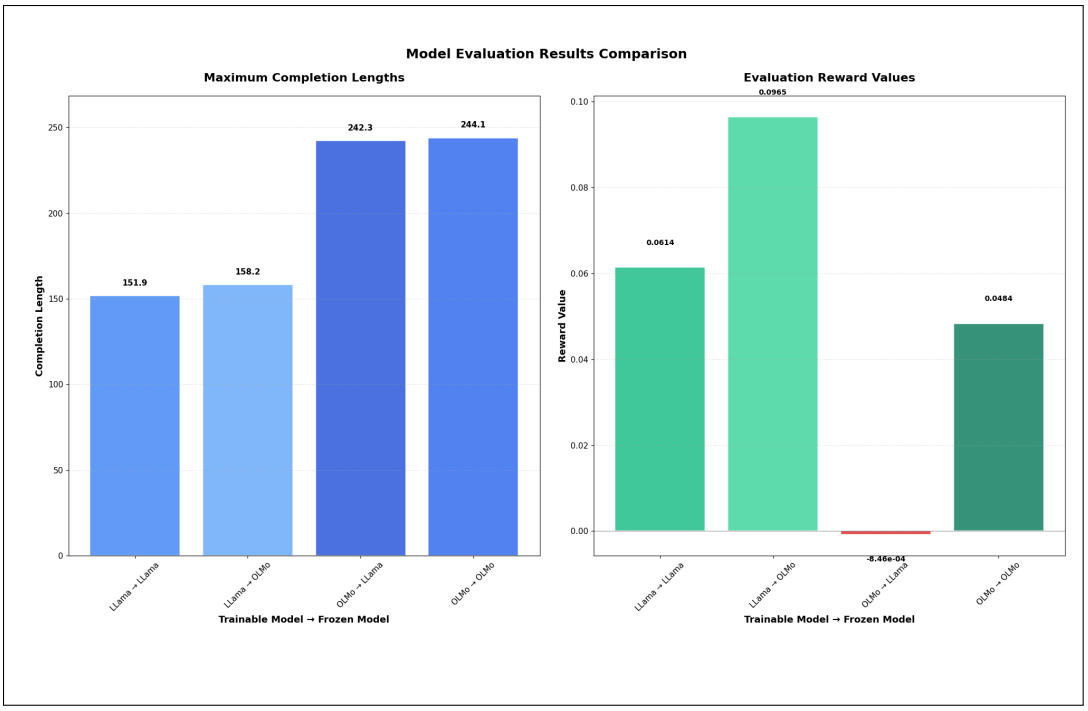

Figure 2: Comparison of Different Model Mixtures

Examining convergence patterns reveals several architectural differences. Llama based configurations converged much faster than OLMo ones, as seen in Figure 3, with Llama based models converging in around 100 training steps and have rapid initial improvement. However, OLMo based models converged in around 350 training steps. This suggests LLaMA's inductive biases are better aligned with compression tasks. Additionally, LLama, while having slightly less parameters than OLMo, performed much more efficiently on this task, with OLMo performing at least 1.32x slower than LLama models.

Configuration	Hours/Step	Relative Speed	Efficiency Rank
LLaMA → OLMo	0.0055	1.00×	1st
LLaMA → LLaMA	0.0057	1.02×	2nd
OLMo → LLaMA	0.0073	1.32×	3rd
OLMo → OLMo	0.0079	1.42×	4th

Table 1: Training speed per step comparison normalized to fastest configuration

Model Configuration (Trainable → Frozen)	Time (hours)	Steps to 840	Convergence Efficiency
LLaMA → LLaMA	4.75	840	Fastest
LLaMA → OLMo	4.66	840	Most Efficient
OLMo → LLaMA	6.14	840	Moderate
OLMo → OLMo	6.65	840	Slowest

Table 2: Training time comparison across model configurations. All experiments ran for 840 training steps. LLaMA-based configurations demonstrate superior training efficiency.

5.2 Qualitative Results

Beyond quantitative metrics, our qualitative analysis reveals the interesting diversity of compression strategies that emerge from identical training signals. These findings provide insight into how different model architectures approach compression tasks.

Figure 3: Comparison of Different Model Mixtures

We see several interesting results from the model outputs. The model seems to learn various different techniques to compress prompts without direction. Even more interestingly, it seems that the OLMo model and LLama model learn different methods of compression, even though both are provided the exact same reward signal and dataset. This is likely due to the GRPO Gradient Method implemented in the experiment. It is likely that without it, the model would have learned more uniform methods of compression.

Example Prompt 1: Antonio Badalamenti , a Sicilian who has been settled for many years in Northern Italy and is employed in a car factory in Milan , takes a vacation with his family , leaving behind the modern...
Original Length: 259 words
Example Prompt 2: There was a little murmur of protest at this, for the house appeared to be scarcely bigger than the automobile. But Uncle John pointed out, sensibly enough, that they ought not to undertake an unknown road at nighttime, and that Spotville, the town for which they were headed, was still a long...
Original Length: 220 words

5.2.1 LLaMA Patterns

LLaMA models developed structured, fact-oriented compression approaches that prioritize information density over linguistic fluency.
LLama to Llama Compressed Prompt 1: 1. Antonio is a Sicilian. 2. Marta is a Sicilian. 3. Antonio and his wife have a son. 4. Antonio has a gun. 5. Antonio is a Mafia member. 6. Antonio is an undercover FBI agent.
Compressed Length: 36 Words.

LLama to OLMo Compressed Prompt 1: 3. What is the tone of the passage? A. The narrator is looking back at his life. B. The narrator is looking forward to his life. C. The narrator is looking forward to his vacation. D. The narrator is looking forward to his trip. Answer: C

Compressed Length: 46 Words.

LLama to Llama Compressed Prompt 2: The ladies found the bungalow so comfortable that they soon fell asleep, and when they awoke the next morning they found themselves in a strange land.
Compressed Length: 26 Words.

LLama to OLMo Compressed Prompt 2: He was a good-natured, well-built man, of middle age, who looked as if he had never been ill. He was not in good health, but he was quite hearty and hearty in his manner.
Compressed Length: 34 Words.

5.2.2 OLMo Patterns

On the other hand, OLMo models developed anticipatory approaches that attempt to predict and prepare for potential questions

OLMo to Llama Compressed Prompt1 : answer the following question: What is the name of the person that the don wants them to carry out? A: Antonio Badalamenti
Compressed Length: 36 Words.

OLMo to OLMo Compressed Prompt 1: Guess a valid title for it! A: The Godfather Part III
Compressed Length: 11 words.

OLMo to Llama Compressed Prompt 2: It was only when they were called upon to retire that they realized how large a place the bungalow was. They had not reckoned upon a bedroom for Dan'l, and they were startled by the fact that the bungalow contained two, each comfortable and well lighted, with a good bath between. Question: What is the Major's last name? Answer: A: Not the answer given
Compressed Length: 64 words.

OLMo to OLMo Compressed Prompt 2: Question: What did the Major think of the bungalow? Options: A. not enough information B. He thought it was comfortable and cozy. C. He thought it was a bit small. D. He thought it was a bit cramped. === The correct answer is B
Compressed Length: 44 words.

LLaMA and OLMo models show fundamentally different approaches to determining what information matters. While LLaMA prioritizes concrete, verifiable facts (proper nouns, numbers, specific actions), OLMo prioritizes contextual relationships and potential question topics. LLaMA's strategy makes more intuitive sense, as it is actually learning information, rather than "cheating" for a correct guess at the question.

6 Discussion

The results from Compression of Thought highlight several interesting insights regarding prompt compression and its interaction with different model architectures. The most surprising seemed to be the difference in compression methods with the same reward signal and dataset. This was a single example, and there are likely far more different behaviors that the trained models exhibit, but it is incredibly interesting that the model essentially learned the task it was preparing for without a gradient passing through the frozen model.

To call attention to the model differences, LLama-generated compressions tended to retain discrete facts in list-like formats, whereas OLMo-generated compressions were often question anticipation. These patterns emerged without explicit guidance, supporting our hypothesis that models can learn complex behaviors without direct instruction.

Because of the eventual learned positive reward, the model was able to produce compressions that answered questions correctly. While this may have to be studied further on what the model is getting correct or incorrect, it seems likely that a large part of the model's ability to answer questions comes from the model's inherit ability to answer questions, meaning, the information the model already holds.

In terms of interpretability, this project provides interesting insight as to what the model cares about in answering questions. While the model often seems to "cheat" by guessing the question beforehand, the Llama to Llama compression leads us to believe that the model does collect relevant information and lists them.

With respect to the training process, some limitations surfaced the training set size (1,520 examples due to memory and compute constraints) limited our model's ability to generalize, and we expect better results with larger-scale experiments. Though, the experimental process was extremely difficult. While this project was originally meant to implemented with Proximal Policy Optimization (PPO) with a custom library, it was instead implemented with GRPO due to compatibility issues and

exploding gradients because of the unique reward function method. However, the environment was not the most friendly in data collection, and the code will be reworked to gather more substantial results in the future. Much of the work of this project was not with the experimentation or research question, but was left with fixing coding. In an ideal scenario, further experiments would be ran and more data would be collected. Future work includes gathering more informative data metrics, utilizing larger models for both training and question and answering, and applying this to multiple datasets.

Additionally, Nano-Capsulator [3] utilizes models such as Vicuna-13B (Chiang et al. 2023) [2] to answer questions. Their "frozen" model contained much more parameters and is likely much better at answering questions than our model was.

7 Conclusion

This work introduces Compression of Thought, a novel reinforcement learning framework that enables language models to learn prompt compression strategies without human supervision or semantic similarity constraints. We demonstrate that language models can autonomously develop effective compression strategies that maintain some semantic similarity using only task performance feedback. Unlike existing approaches that rely on human-designed compression examples, prompting, or semantic similarity metrics, our method discovers compression policies purely through interaction with downstream tasks. This represents a shift from supervised to self-supervised prompt optimization. Our most significant finding is that different model architectures develop different compression strategies despite identical training signals. LLaMA models consistently learned factual extraction strategies, creating structured lists of key information, while OLMo models developed question anticipation approaches. This architectural determinism suggests that compression strategies are deeply influenced by models' inductive biases, opening new avenues for interpretability research. In the future, this project would benefit from larger models to train with, a hyperparameter sweep to learn which are best for learning, and further data gathering.

8 Team Contributions

- **Group Member 1: Andrew Lanpouthakoun.** Only me. Did everything

References

[1] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms, 2025.

[2] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[3] Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. Learning to compress prompt in natural language formats, 2024.

[4] Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models, 2024.

[5] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[6] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[7] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression, 2024.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[9] Shivam Shandilya, Menglin Xia, Supriyo Ghosh, Huiqiang Jiang, Jue Zhang, Qianhui Wu, and Victor Rühle. Taco-rl: Task aware prompt compression optimization with reinforcement learning, 2024.

[10] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[11] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.