Improving Chain of Thought with Chain of Shortcuts

Satchel Grant
Department of Psychology
Stanford University
Stanford, CA 94305
grantsrb@stanford.edu

Sahil Kulkarni
Microsoft Corporation
Redmond, WA 98052

Noah Goodman
Department of Psychology and Computer Science
Stanford University
Stanford, CA 94305

Abstract

Large Language Models (LLMs) have demonstrated remarkable language modeling and sequence modeling capabilities with capabilities like In-Context Learning (ICL) and Chain of Thought (CoT) reasoning, akin to human working memory and reasoning. Drawing inspiration from dual process theory in human cognition, we propose a novel training technique called Chain of Shortcuts (CoS) that bridges the gap between LLMs' System 1 (automatic) and System 2 (deliberate) modes. CoS enables LLMs to compress reasoning trajectories, encouraging associations between earlier and later steps in problem-solving, resulting in shorter, more flexible solutions. We demonstrate that CoS-trained language models maintain or outperform baseline models while generating distilled problem solutions, enhancing stability during training, and excelling in high-temperature environments. CoS's effectiveness increases with the number of transformer layers until saturation. Our work not only contributes to mathematical transformers but also offers insights into human dual process theory, paving the way for more efficient and robust AI systems.

1 Introduction

Large Language Models (LLMs) have been shown to be powerful language modeling tools Brown et al. (2020a); Bommasani et al. (2021); Kaplan et al. (2020) and powerful sequence modeling tools in many other domains Fan et al. (2022); Tran and Soleymani (2022); Wang et al. (2021). They have been shown to have a remarkable capacity for learning patterns and tasks in context, without gradient updates, in a process known as In-Context Learning (ICL) Garg et al. (2022); Brown et al. (2020a). And when given the ability to reason through a problem in a process called Chain of Thought (CoT), LLMs can perform significantly better than when attempting to solve the task directly Wei et al. (2022b); Zhou et al. (2022a); Kojima et al. (2022); Nye et al. (2021).

There are striking similarities between LLM's ICL and CoT to human working memory and mental reasoning. Humans, for example, have the ability to quickly learn from recent experience, and humans often benefit from multi-step mental reasoning and using external tools like pencil and paper to solve problems. Many papers have demonstrated similarities between the ways in which LLMs and humans reason (Dasgupta et al., 2022; Lampinen et al., 2022), and many advances in Artificial Intelligence (AI) have been inspired by biological intelligence (Hassabis et al., 2017).

Preprint. Under review.

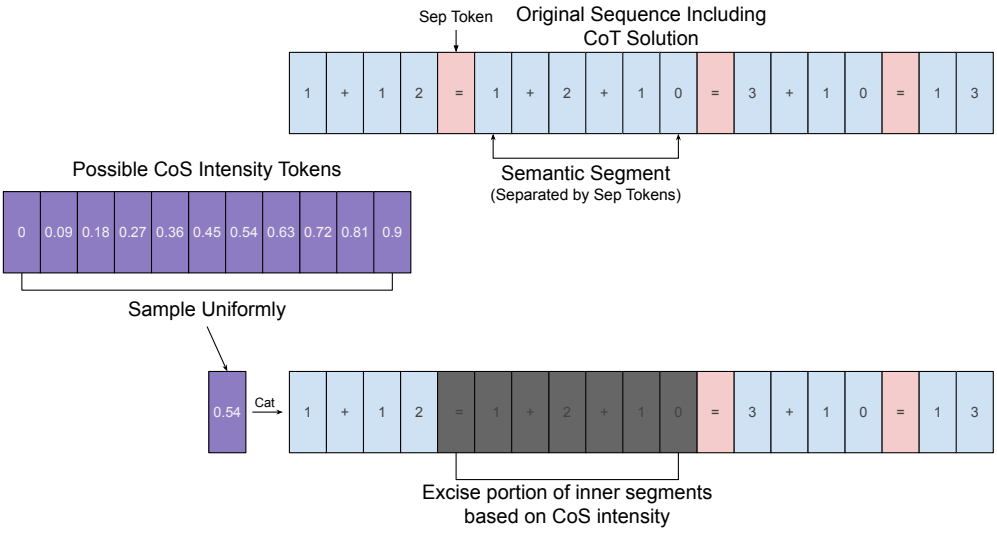

Figure 1: An illustration of Chain-of-Shortcuts (CoS). A CoS intensity token is sampled uniformly from a per-determined set of intensity tokens. The intensity token indicates the proportion of steps to be dropped from the original sequence. The intermediate steps are selected to be approximately equally spaced between the initial problem and the final solution.

Taking a similar line of inspiration from human intelligence, we base our method on Dual Process Theory (DPT) from human cognition literature. DPT suggests that humans have two systems of thought: a fast, automatic, unconscious mode (System 1), and a slow, deliberate, conscious mode (System 2) Evans (1990); Wason and Evans (1974); Evans and Stanovich (2013); Kahneman (2011). In the context of language models, single token generation (i.e. a single forward pass through a transformer) may be viewed as analogous to System 1. Multi-step CoT-like reasoning can be viewed as analogous to System 2. We argue that for choices at each iteration to be robust for problem solving, one requires the System 2 process to make informed use of the computational primitives provided by System 1. And, perhaps, distilling the deductive thoughts generated by System 2 into System 1 primitives could enable the system to perform longer and more complicated reasoning. Perhaps, also, shifting computations from System 2 to System 1 could provide robustness to a model's reasoning by offering redundant pathways for the same computation and by reducing opportunities for mistakes in cases of high temperature token sampling.

In an effort to explore these ideas, we introduce a new training technique called Chain of Shortcuts (CoS). CoS is a procedure that can be added to any LLM training that has data with intermediate steps. CoS can also be applied as a self-supervised technique to existing LLMs for compressing reasoning trajectories into fewer steps. CoS consists of removing intermediate segments of the context during training. This encourages the transformer to form associations between earlier steps in a reasoning trace with later steps, enabling the transformer to generate shorter solutions for a given problem, and giving it more ways to perform an otherwise deterministic reasoning trace.

We show that language models using CoS are at least as performant as models trained without CoS (in many cases out-performing the baselines), they learn to generate distilled problem solutions with a relatively small decrease in performance, and they have more stable performance over the course of training. We also show that CoS models perform better than baselines in high temperature environments, and we demonstrate that CoS improves with increasing number of transformer layers until performance saturation. We hope this work will serve as a stepping stone for improving mathematical transformers and understanding human dual process theory.

2 Related Work

A number of approaches have been proposed to address the quadratic complexity with sequence length of transformers. In a review from Tay et. al. 2020, they broke up the methods into six categories: Recurrence, Memory/Downsampling, Learnable Patterns, Sparsity, Fixed/Factorized/Random Patterns, Low Rank/Kernels (Tay et al., 2020). Approaches that are relevant to CoS are the Compressive Transformer (Rae et al., 2019), which applies convolutions over portions of the context window in order to create compressed representations. This is similar in the sense that CoS can be viewed as a compression of CoT traces. Similarly, our method introduces new, untrained CoS intensity tokens to to the token sequence. This is reminiscent of Prompt Tuning (Lester, Al-Rfou, and Constant, 2021; He et al., 2022) where they introduce and train new tokens that are prepended to a given context. This is also similar to Gist Tokens from Mu, Li, and Goodman (2023) and context compression (Chevalier et al., 2023; Grant and Kulkarni, 2023).

Many works have shown the benefits of allowing transformers to use thought traces to solve math problems. An early work that inspired the generation of our math sequences was Scratchpads, where Nye et al. (2021) trained a transformer to solve math problems by training it to perform a series of algorithmic steps. Since then some works have shown that such algorithmic approaches can be learnable from the context (Zhou et al., 2022b). A more recent work has shown that it is possible for LLMs to solve very large and complicated math problems using a step-by-step procedure. Yang et al. (2023) showed that with sufficient data, sufficient model size, and curriculum learning, they could get a transformer to solve various multiplication, addition, subtraction, and decimal problems up to 12 digits long.

Another recent work worth mentioning is from Adams et al. (2023) who showed improvements to GPT-4's summaries when prompted to give varying levels of summary density. This relates in that there can be benefits from modeling a sparser/denser thought trace depending on the particular task at hand. And even more recently, Zhang and Parkes (2023) showed that iteratively bootstrapping solutions to new problems using CoT and then training the LLM to map the initial problem to the final solution without CoT enabled them to train a model to perform arithmetic on models as long as 25 digits. This can be seen as a specific instance of CoS, in which the shortcut intensity has two possible values, 0 and 1.

3 Methods

For the purpose of this discussion we restrict our view to a single domain of autoregressive, next-token-prediction problems. We note, however, that Chain of Shortcuts (CoS) naturally extends to multiple domains within sequence based modeling. We begin with a set of problems from domain $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{s}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$, where the input $\mathbf{x}^{(i)}$ is a sequence containing the problem statement, i.e., $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_m^{(i)})$, the input $\mathbf{s}^{(i)}$ is a sequence of intermediate reasoning steps (also referred to as *semantic steps*) $\mathbf{s}^{(i)} = (s_1^{(i)}, \ldots, s_m^{(i)})$ where each step $s_j^{(i)}$ is a sequence of tokens containing intermediate information used to solve the problem, and the output $\mathbf{y}^{(i)}$ is a sequence of tokens containing the answer. Each sequence of steps, $\mathbf{s}^{(i)}$ provides a comprehensive deductive path motivating the final answer $\mathbf{y}^{(i)}$. We note that certain domains are better suited than others for the creation of such a set $\mathbf{s}^{(i)} \in \mathcal{S}$. Therefore, to prove principle, we restrict ourselves to the domain of relatively simple addition and multiplication problems.

3.1 Chain of Shortcuts

CoS introduces a set of tokens $\mathcal{C} = \{c_0, c_1, \ldots, c_k\}$ to a sequence model that are used to indicate the proportion of intermediate reasoning steps that will be excised from the sequence. During training, an intensity token, $c_j^{(i)}$, is sampled for each training sequence in the batch of training samples and is then prepended to the sequence. The intensity token and the original sequence length are used to determine the number of intermediate semantic steps, $s_j^{(i)}$ to excise. The dropped semantic steps are selected such that they are approximately evenly spaced among the original sequence. This results in a batch of samples where each sample takes the form $d^{(i)} = \{c_j^{(i)}, x_1^{(i)}, \ldots, x_\ell^{(i)}, s_1^{(i)}, \ldots, s_m^{(i)}, y_1^{(i)}, \ldots, y_m^{(i)}\}$ and some subset of $\mathbf{s}^{(i)}$ is removed from the sequence. We perform this efficiently by masking the

shortcutted intermediate segments and shifting the positional indices accordingly. When using the model for inference, the CoS tokens can be selected based on the desired amount of shortcutting.

3.2 Model and Training Details

We begin this section by noting that CoS, much like CoT, can in principle be used with many sequence based model architectures. We, however, limit this work to transformer models (Vaswani et al., 2017). More specifically, we trained models from scratch, creating the architectures from the Huggingface python package using LlamaConfig configurations (Wolf et al., 2019; Touvron et al., 2023). Llama based models are different from the original transformer architecture in that they pre-normalize the input to each transformer layer (Brown et al., 2020b), they use a SwiGLU activation function (Shazeer, 2020), and they use rotary positional embeddings RoPE (Su et al., 2022). We used an embedding and hidden size of 128, an intermediate size of 512, and 4 attention heads for each transformer layer. We explore the effect of varying the number of layers on performance in the experiments section of this work. We used the default settings for all other options.

We used batch stochastic gradient descent to train the models to autoregressively predict the next token in the sequence. We removed the initial problem statement from the prediction task, including it only in the context to be used for next token prediction. The first predicted token always came just after the first equals sign following the intitial problem. During training, we used *teacher forcing*, in the sense that we used the ground truth, rather than the model's predictions, as the input to the next step in the training sequence. For all inference tasks, we sampled the model's predictions as the input for the next step. We explore the effect of different sampling temperatures in the experiments section.

We used an Adam optimizer (Kingma and Ba, 2017), a batch size of 128, label smoothing of 0.15, and a learning rate with 2500 linear warmup steps followed by a learning rate decay according to the equation $\alpha * \frac{w^\gamma}{step^\gamma}$ where α is the maximum learning rate, w is the number of warmup steps, $step$ is the current update step, and γ is a manually tuned decay parameter. We set $\alpha = 0.0045$ and $\gamma = 0.1$ for all experiments unless otherwise stated. We used PyTorch for all autodifferentiation (Paszke et al., 2019).

3.3 Math Environment

Each training sequence corresponds to a tokenized version of a math problem. The math problems consist of addition and multiplication. In this work, we focus on problems that contain 2-3 starting values in which the values can range from 0-5000 when being summed and 0-7 when being multiplied. These numbers were chosen based on ensuring the performance ceiling was not 100% and based on available GPU capacity. The intermediate reasoning steps were generated according to the following steps.

1. **Rearrange:** order the entities such that multiplied terms are are first, followed by numbers sorted by Order of Magnitude (OoM) with smaller OoMs first and larger OoMs last. Perform all rearrangements in a single step.

2. **Decompose:** decompose the numbers into their primitive components. Multiplication terms are handled first, separating out the terms into a sum of individual values, taking a single step for each decomposition. Then non-multiplied terms are separated into their respective OoMs. This only occurs if there are multiple quantities of that particular OoM (i.e. if there are at least two numbers with non-zero values in the one's place, separate those numbers into a sum of the ones place and the rest of the number. Do this for all OoMs). Each individual number decomposition takes up a single intermediate step.

3. **Combine Same OoM:** select two numbers with the smallest OoM and sum them together (these should be the leftmost numbers). If their sum results in a number with a greater OoM, then return to step 2. Repeat until only one number exists at the current OoM.

4. **Combine Different OoMs:** select the remaining number with the smallest OoM and sum it with the leftmost number of the next possible OoM. Return to step 3 and repeat until only a single number remains.

We note that the algorithm can be made much simpler if we were to perform the Combine Different OoMs as a final step, only when all OoMs have one number remaining. We will make this change in future work.

All models were trained with 10,000 math problems sampled at the beginning of training. The models were evaluated on 5,000 samples, different than those in the training set, also generated at the beginning of training. We chose the quantity 10,000 empirically to maintain a relative level of performance consistency without reaching the performance ceiling of 100% accuracy.

Problem	**Solution with Intermediate Steps**
$3 \times 2 + 18 =$	$1 \times 3 + 3 + 18 = 3 + 3 + 18 = 3 + 3 + 8 + 10 = 6 + 8 + 10 =$ $14 + 10 = 24$
$4 + 17 + 8 =$	$8 + 4 + 17 = 7 + 8 + 4 + 10 = 4 + 15 + 10 = 5 + 4 + 10 + 10 =$ $9 + 10 + 10 = 19 + 10 = 29$
$802 + 203 =$	$2 + 800 + 203 = 3 + 2 + 800 + 200 = 5 + 800 + 200 = 805 + 200 = 1005$

Table 1: Examples of generated problems and solutions.

4 Experiments

4.1 Model Variants

- **CoS:** models trained with variable intensities of shortcutting on sequences with intermediate semantic steps.
- **Vanilla:** models trained without shortcutting on sequences with intermediate semantic steps.
- **Direct:** models trained to map problems to solutions directly, without any intermediate steps.

4.2 Main Comparison

We first compare CoS to an upper baseline set by the performance of the same model architecture trained to produce the full output sequence without any shortcutting (the Vanilla variant). This is equivalent to CoS with a shortcutting intensity of 0. We also show the results of the same model architecture trained to map problems to answers directly, without intermediate steps (the Direct variant). We can see from Figure 2 that the performance of the CoS variant at least matches that of the Vanilla variant at lower CoS intensities. (See Size Effects for why we are hesitant to say CoS performance surpasses that of the Vanilla). We can also see that the CoS performance remains relatively unaffected even when it removes large portions of tokens from its solution trace. There is a performance drop at large CoS intensities, that remains higher performance than the Direct mapping baseline for a range of temperatures. We see this performance gap as evidence in favor of the view that CoS provides some form of regularization or curriculum learning that enables models to learn direct mapping solutions better than naive, direct mapping training solutions.

4.3 Size Effects

We can see that model size is potentially important to the performance of CoS variants from Figure 3. At 2 layers, the Vanilla variant appears to perform better than the CoS variant. With more layers, the Vanilla and CoS variants perform approximately equally across a range of model sizes. To qualify the results in this work, we feel obligated to note that some of the performance curves in Figure 3 have the appearance that they could be improved from further hyperparameter tuning. We say this due to the early performance volatility in multiple panels and the strong performance of the 2 layer Direct variant relative to the larger Direct variants. We used the same hyperparameters for all model trainings. We note that we did try a range of hyperparameter settings before settling on the parameters used in these experiments. The space of possible combinations is large, however, and has the potential to change results. We leave a more extensive exploration of hyperparameter space to future work.

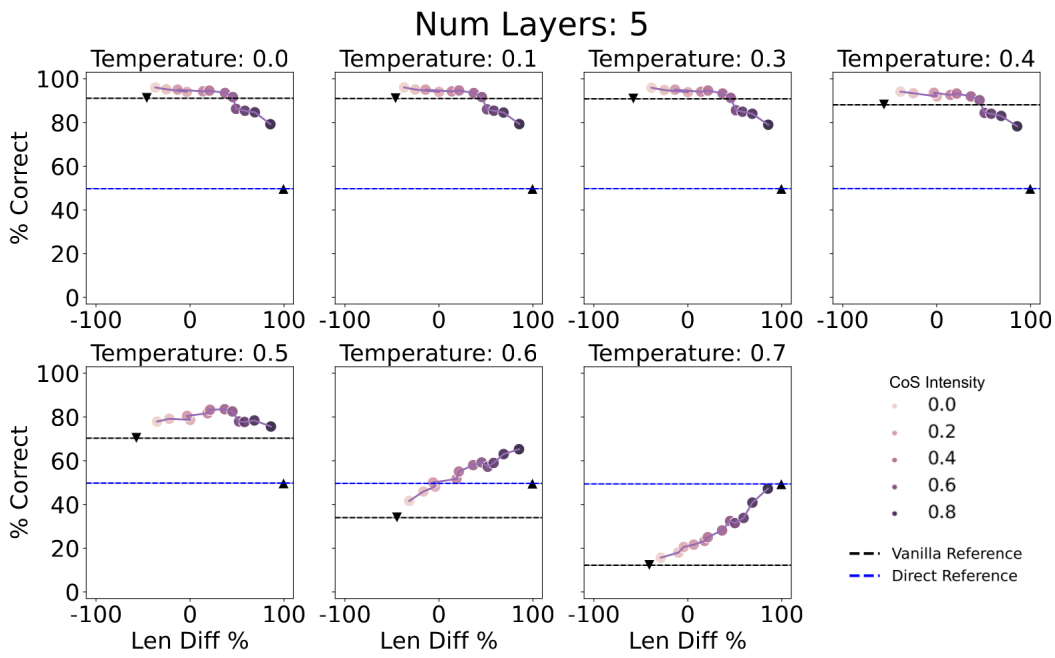

Figure 2: Accuracy of the final answer as a function of predicted solution length over different sampling temperatures for models with 5 layers. The black, upside down triangle shows the performance of the Vanilla baseline model. The black dashed line provides a visual reference to this performance. The black right-side up triangle shows the performance of the Direct mapping baseline model. The dashed blue line provides a visual reference to this performance. The purple gradient points show the performance of the CoS model at different CoS intensities. We can see that larger CoS intensities decrease the solution length while maintaining different performance levels. We can also see that larger CoS intensities perform better at higher temperatures.

4.4 Temperature Effects

We see two potential benefits of a CoS like approach to reasoning. The first is a resource rational argument: shorter solutions demand less computation. The second is a sampling argument, that in stochastic token selection, shorter solutions provide fewer opportunities for mistakes. We have already seen evidence of the first benefit from the solution lengths demonstrated in Figure 2. We can also see from Figure 2 that the sampling argument is demonstrated in the CoS models. At higher sampling temperatures, the performance of the shorter solutions is less impacted than the longer solutions. The CoS variants have a clear performance advantage over the Vanilla variants at higher temperatures. Interestingly, the Direct mapping performance is completely unaffected at the sampling temperatures explored in this work.

4.5 Out of Distribution Generalization

We hypothesized that learning a range of solution lengths and a wider range of primitive computations would enable the CoS models to better generalize to problems containing quantities beyond that of the training distribution. We can see the performance on these Out of Distribution (OoD) samples in Figure 4. From the figure, we can see that a similar performance relationship holds the OoD samples as with the within distribution samples.

4.6 Training Volatility

Lastly, we analyze the training volatility of the different variants. We can see from Figure 5 that the CoS variants have the a more stable performance trajectory than the Vanilla variants averaged over the course of training across a a range of model sizes. This metric suffers from the same potential

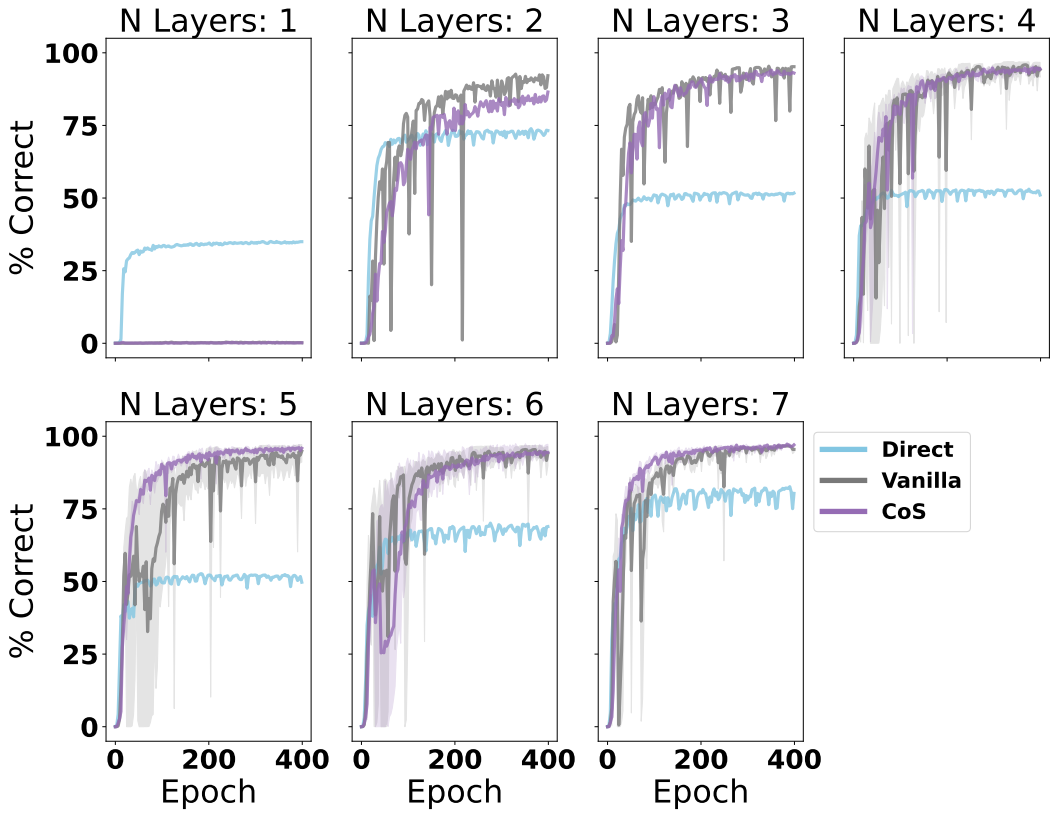

Figure 3: Accuracy of the final answer as a function of training epoch over different model sizes using a sampling temperature of 0 (argmax over logits). At the time of writing this paper, we used 3 seeds for the CoS and Vanilla variants at layer counts of 4, 5, 6, and 7. And we used 1 seed for all other curves. Error bars are Standard Error over seeds. We can see that the CoS models perform at least as well as the Vanilla models at sufficient layer counts.

issue as mentioned in the section on Size Effects, that hyperparameter choices can potentially change results. We saw similar volatility trends across a range of hyperparameter choices (see supplemental figures), however, we leave a more complete exploration of the space of hyperparameters to future work.

5 Discussion/Conclusion

This work has shown that CoS works as a method for improving LLM reasoning efficiency. CoS variants were often able to reduce solution lengths by more than 50% with a negligible impact on performance. We also see that CoS improves LLM reasoning ability in relatively high temperature settings. These results encourage speculation when viewed as an analogy to Dual Process Theory in humans. Under the assumption that there is a benefit from shifting computations from System 2 to System 1, perhaps the potential benefits should not be viewed as mutually exclusive. The dual processes in human reasoning potentially provide both resource efficiency and improvements to reasoning sampling.

As a training technique CoS appears to provide benefits to training stability and potentially provides benefits to generalization. Other, recent work has shown the benefits of using CoS to bootstrap new data from a CoT reasoning process and consolidate the learned solutions into direct mappings through CoS with an intensity of 1 (Zhang and Parkes, 2023). Thus, we look forward to exploring the benefits of CoS as an unsupervised learning algorithm in future work. Our results are currently insufficient to draw strong conclusions on performance comparisons between CoS and Vanilla training variants

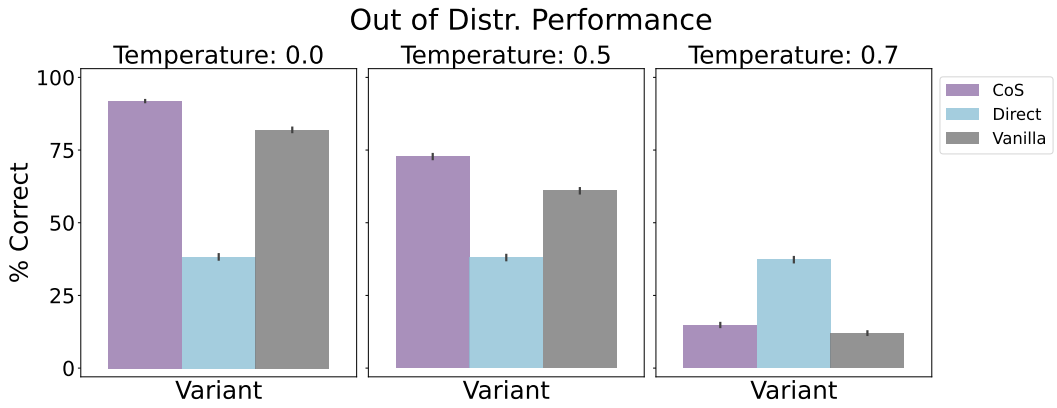

Figure 4: Accuracy of the final answer using the lowest CoS intensity and models with 5 layers on problems containing at least one value in the range 5001-6000 (1000 more than the largest number in the training distribution). Error bars are Standard Error over evaluation samples. We can see that the Out of Distribution (OoD) performance has a similar trend to within distribution samples when comparing to Figures 2 and 3.

(due to under-exploration of the hyperparameter space and limited seeds). CoS, however, appears to be at least as good as Vanilla baselines while providing the improvements already mentioned.

We found it interesting that size had an impact on relative performance of the different variants. It would appear that the models need a minimum capacity, or a minimum number of layers, to learn the wider collection of primitive computations induced in CoS training variants. This is potentially related to LLM scaling laws and emergent abilities (Kaplan et al., 2020; Brown et al., 2020b; Wei et al., 2022a). We leave exploration of this potential connection to future work.

Acknowledgments and Disclosure of Funding

Thank you to the Parallel Distributed Processing Lab for funding access to the CCN Cluster for model trainings. And thank you to the 2023 HAI-Google Cloud Credits Grant Program for also funding part of this research.

References

Adams, G.; Fabbri, A.; Ladhak, F.; Lehman, E.; and Elhadad, N. 2023. From Sparse to Dense: GPT-4 Summarization with Chain of Density Prompting. arXiv:2309.04269.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R. B.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; Brynjolfsson, E.; Buch, S.; Card, D.; Castellon, R.; Chatterji, N. S.; Chen, A. S.; Creel, K.; Davis, J. Q.; Demszky, D.; Donahue, C.; Doumbouya, M.; Durmus, E.; Ermon, S.; Etchemendy, J.; Ethayarajh, K.; Fei-Fei, L.; Finn, C.; Gale, T.; Gillespie, L.; Goel, K.; Goodman, N. D.; Grossman, S.; Guha, N.; Hashimoto, T.; Henderson, P.; Hewitt, J.; Ho, D. E.; Hong, J.; Hsu, K.; Huang, J.; Icard, T.; Jain, S.; Jurafsky, D.; Kalluri, P.; Karamcheti, S.; Keeling, G.; Khani, F.; Khattab, O.; Koh, P. W.; Krass, M. S.; Krishna, R.; Kuditipudi, R.; and et al. 2021. On the Opportunities and Risks of Foundation Models. *CoRR*, abs/2108.07258.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020a. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.;

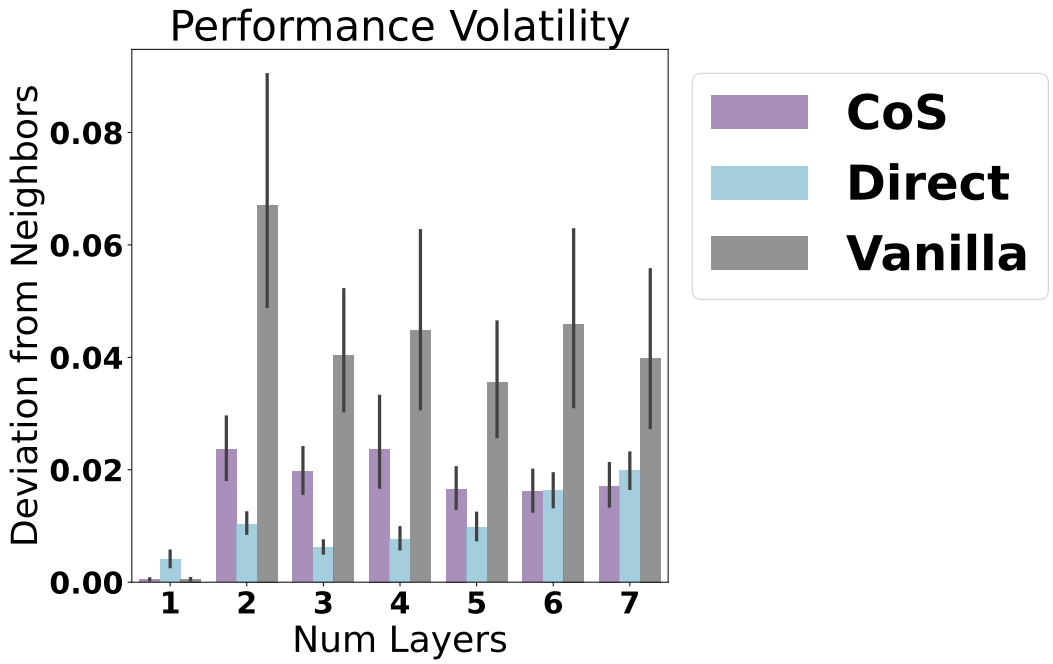

Figure 5: Volatility of final answer accuracy as a function of the number of transformer layers averaged over the course of all training epochs. Volatility was calculated as the deviation of the performance for a given epoch from the mean performance of the two surrounding epochs. Larger deviations indicate lower training stability and increased performance variance. Error bars were calculated as standard error over all possible epochs. 3 model seeds were used for the CoS and Vanilla variants with layer counts of 4, 5, 6, and 7. 1 model seed was used for all other data.

Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020b. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.

Chevalier, A.; Wettig, A.; Ajith, A.; and Chen, D. 2023. Adapting Language Models to Compress Contexts. arXiv:2305.14788.

Dasgupta, I.; Lampinen, A. K.; Chan, S. C. Y.; Creswell, A.; Kumaran, D.; McClelland, J. L.; and Hill, F. 2022. Language models show human-like content effects on reasoning. arXiv:2207.07051.

Evans, J.; and Stanovich, K. E. 2013. Dual-Process Theories of Higher Cognition. *Perspectives on Psychological Science*, 8: 223 – 241.

Evans, J. S. B. T., ed. 1990. *Bias in Human Reasoning: Causes and Consequences*. Psychology Press.

Fan, L.; Wang, G.; Jiang, Y.; Mandlekar, A.; Yang, Y.; Zhu, H.; Tang, A.; Huang, D.-A.; Zhu, Y.; and Anandkumar, A. 2022. MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge.

Garg, S.; Tsipras, D.; Liang, P.; and Valiant, G. 2022. What Can Transformers Learn In-Context? A Case Study of Simple Function Classes.

Grant, S.; and Kulkarni, S. 2023. Leveraging Large Language Models for Context Compression.

Hassabis, D.; Kumaran, D.; Summerfield, C.; and Botvinick, M. 2017. Neuroscience-Inspired Artificial Intelligence. *Neuron*, 95(2): 245–258.

He, Y.; Zheng, H. S.; Tay, Y.; Gupta, J.; Du, Y.; Aribandi, V.; Zhao, Z.; Li, Y.; Chen, Z.; Metzler, D.; Cheng, H.-T.; and Chi, E. H. 2022. HyperPrompt: Prompt-based Task-Conditioning of Transformers.

Kahneman, D. 2011. *Thinking, fast and slow*. New York: Farrar, Straus and Giroux. ISBN 9780374275631 0374275637.

Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling Laws for Neural Language Models. *CoRR*, abs/2001.08361.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large Language Models are Zero-Shot Reasoners.

Lampinen, A. K.; Dasgupta, I.; Chan, S. C. Y.; Matthewson, K.; Tessler, M. H.; Creswell, A.; McClelland, J. L.; Wang, J. X.; and Hill, F. 2022. Can language models learn from explanations in context? 1–29.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning.

Mu, J.; Li, X. L.; and Goodman, N. 2023. Learning to Compress Prompts with Gist Tokens. arXiv:2304.08467.

Nye, M.; Andreassen, A. J.; Gur-Ari, G.; Michalewski, H.; Austin, J.; Bieber, D.; Dohan, D.; Lewkowycz, A.; Bosma, M.; Luan, D.; Sutton, C.; and Odena, A. 2021. Show Your Work: Scratchpads for Intermediate Computation with Language Models. arXiv:2112.00114.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopt, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 8024–8035.

Rae, J. W.; Potapenko, A.; Jayakumar, S. M.; and Lillicrap, T. P. 2019. Compressive Transformers for Long-Range Sequence Modelling.

Shazeer, N. 2020. GLU Variants Improve Transformer. arXiv:2002.05202.

Su, J.; Lu, Y.; Pan, S.; Murtadha, A.; Wen, B.; and Liu, Y. 2022. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864.

Tay, Y.; Dehghani, M.; Bahri, D.; and Metzler, D. 2020. Efficient Transformers: A Survey.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

Tran, M.; and Soleymani, M. 2022. A Pre-trained Audio-Visual Transformer for Emotion Recognition. *CoRR*, abs/2201.09165.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, R.; Chen, D.; Wu, Z.; Chen, Y.; Dai, X.; Liu, M.; Jiang, Y.; Zhou, L.; and Yuan, L. 2021. BEVT: BERT Pretraining of Video Transformers. *CoRR*, abs/2112.01529.

Wason, P.; and Evans, J. 1974. Dual processes in reasoning? *Cognition*, 3(2): 141–154.

Wei, J.; Tay, Y.; Bommasani, R.; Raffel, C.; Zoph, B.; Borgeaud, S.; Yogatama, D.; Bosma, M.; Zhou, D.; Metzler, D.; Chi, E. H.; Hashimoto, T.; Vinyals, O.; Liang, P.; Dean, J.; and Fedus, W. 2022a. Emergent Abilities of Large Language Models. arXiv:2206.07682.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Chi, E. H.; Le, Q.; and Zhou, D. 2022b. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *CoRR*, abs/2201.11903.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; Davison, J.; Shleifer, S.; von Platen, P.; Ma, C.; Jernite, Y.; Plu, J.; Xu, C.; Scao, T. L.; Gugger, S.; Drame, M.; Lhoest, Q.; and Rush, A. M. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing.

Yang, Z.; Ding, M.; Lv, Q.; Jiang, Z.; He, Z.; Guo, Y.; Bai, J.; and Tang, J. 2023. GPT Can Solve Mathematical Problems Without a Calculator. arXiv:2309.03241.

Zhang, H.; and Parkes, D. C. 2023. Chain-of-Thought Reasoning is a Policy Improvement Operator. arXiv:2309.08589.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q.; and Chi, E. 2022a. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models.

Zhou, H.; Nova, A.; Larochelle, H.; Courville, A.; Neyshabur, B.; and Sedghi, H. 2022b. Teaching Algorithmic Reasoning via In-context Learning. arXiv:2211.09066.