A Pipeline for Assessing Metacognitive Reasoning in Energy-Based Transformers while Generating Code

Tailia Malloy, Prateek Kumar Rajput, Iyiola E. Olatunji, Jacques Klein, Tegawende F. Bissyande
University of Luxembourg
Trustworthy Software Engineering (TruX) Group
{tailia.malloy, prateek.rajput, emmanuel.olatunji, jacques.Klein, tegawende.bissyande}@uni.lu

Abstract

While LLMs excel at generating code snippets, they often fall short in agentic programming scenarios that require sustained, iterative interaction with a development environment. This shortcoming persists even in massive-scale models with explicit reasoning capabilities such as Claude Sonnet-4.5-Thinking and GPT-5-Codex, suggesting a fundamental gap in metacognitive capabilities: the ability to self-assess, deliberate, and correct course during a complex task. Energy-Based Transformers (EBT) have recently been proposed as a method of improving scaling performance on complex tasks that require thinking and deliberation. EBTs perform multiple forward passes at inference while modeling uncertainty and validating predictions, in an attempt to mimic human System 2 thinking, which is related to deliberate and effortful metacognitive reasoning. In this work we propose a framework for assessing the metacognitive reasoning capabilities of EBTs for code generation tasks. Our core contribution is a feedback-aware inference mechanism that dynamically scales the model's "thinking depth" (number of forward passes) based on external feedback. Our proposed framework supports an optional human-in-the-loop mode where code engineers can interact with the model and control the degree of forward pass reasoning through feedback. This proposed approach has the potential to allow LLMs to interact with human programmers as another member of the code engineering team, and dramatically improve the speed and quality of human generated code by interacting with an LLM agent.

1 Introduction

In human psychology, metacognition is a notoriously difficult concept to concretely define and measure [2, 17, 38, 37], with significant debate over whether or not Artificial Intelligence (AI) models will ever be capable of metacognition as it exists in humans. Some research argues against this position [34, 3], and other research argues in favor [7, 16]. Outside of the debate over whether AI systems could theoretically exhibit features of higher level consciousness like metacognition, there is also considerable discussion over whether or not current AI systems already exhibit this type of reasoning. Again, with some arguing in favor [6, 28, 12, 20], and others arguing against [5, 36, 44].

One common feature of metacognition as it is described in psychology and cognitive science is the process of reasoning over one's thought processes [18, 1], such as by assessing what information is known and what is not known[45], or what previously reasoned pieces of information are incorrect [39]. Generative Artificial Intelligence (GAI) models have been proposed as one method to equip AI systems with metacognitive reasoning [11, 50], using methods such as Large Language Models

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: EurIPS Workshop on Metacognition in Generative AI.

(LLMs) to repeatedly take in as input previous model outputs [30], in an attempt reason over what information is and is not known by the model [11].

Using LLMs with hundreds of billions to trillions of parameters can enable these systems to generate output while performing reasoning over previously generated information [48]. However, it can be difficult for models that reason over their previous outputs to determine when information they previously generated is in fact incorrect [47, 29, 25]. As a result, reasoning LLMs still lag behind humans in tasks that require reasoning over their thought processes while interacting with complex environments, such as when writing code in an agentic setting [49] or assessing long inputs of code that may have errors [43], making current LLMs difficult to integrate directly on programming teams as an independent agent. A related finding within the domain of human identification of AI generated code has shown that AI written code is easier to identify as malicious compared to code written by humans [32].

Energy based transformers have been proposed as one method of achieving so-called 'System 2' thinking [20], or the slow and deliberate thinking that humans perform when task demands complex and thoughtful reasoning [24, 14, 23, 15], which has been related to metacognition [42, 33, 41, 8]. If EBTs can truly exhibit system 2 thinking, they should be able to produce improved metacognitive reasoning such as accurately reasoning over previous output through a deep understanding of uncertainty, thoughtfully validating their previous and subsequent outputs, and dynamically allocating additional computational resources to outputs that are identified as being highly relevant.

In this work, we present a method for assessing the metacognitive reasoning capabilities of EBTs. This is done to answer the following research questions:

RQ1. Do scaling improvements of EBTs extend to agentic code generation, and if so why.

RQ2. Do the proposed features of EBTs exhibit metacognitive reasoning when generating code.

RQ3. Can feedback on code generation be used to improve outputs at inference time with EBTs.

However, it is important to note that this work is merely an outline of the process of assessing EBTs for metacognitive capabilities in software generation. We do not perform experiments testing the effectiveness of our proposed approach because we hope to develop our reasoning assessment method more thoroughly before performing costly experimentation.

2 Metacognitive Reasoning Assessment

2.1 EBT Algorithms

Algorithm 1: TRAINING

Input: Context x, Target y, EBM $E_\theta(x, \hat{y})$
Hparams: Steps N, Step Size α, Loss $J(\cdot)$

Sample $\hat{y}_0 \sim \mathcal{N}(0, I)$;
for $i = 0, \ldots, N-1$ **do**
  $\hat{y}_{i+1} \leftarrow \hat{y}_i - \alpha \nabla_{\hat{y}_i} E_\theta(x, \hat{y}_i)$;
$\mathcal{L} \leftarrow J(\hat{y}_N, y)$;
return $\mathcal{L}$, update E_θ;

Algorithm 2: INFERENCE

Input: Context x, EBM $E_\theta(x, \hat{y})$
Hparams: Steps N, Step Size α, Samples M

for $j = 1, \ldots, M$ **do**
  Sample $\hat{y}_{0,j} \sim \mathcal{N}(0, I)$;
  for $i = 0, \ldots, N-1$ **do**
    $\hat{y}_{i+1,j} \leftarrow \hat{y}_{i,j} - \alpha \nabla_{\hat{y}_{i,j}} E_\theta(x, \hat{y}_{i,j})$;
return $\hat{y}^* = \text{argmin}_j \, E_\theta(x, \hat{y}_{N,j})$;

The EBT training algorithms above are replicated from [20], to allow us to describe the features of EBT models that make them useful for testing metacognitive reasoning. Algorithm 1 details how EBTs are trained to predict the output y given the context x by learning to minimize the energy $E_\theta(x, y)$ which correlates to the inverse of the likelihood of the output with respect to the input. At inference time Algorithm 2 is used to allow for dynamic allocation of optimization of the energy minimization process, allowing for flexibility in the amount of compute resources given to determining any given output relative to the input.

2.2 Coding with EBTs

Three important features of EBTs make them a good target for our proposed method of assessing metacognitive reasoning in code generation. These are Dynamic Compute Allocation, Modeling

Uncertainty and Prediction Verification. In the remainder of this section we describe each of these model features and how they are related to our goal of assessing metacognitive reasoning in code generation. After this, we describe how we can use EBTs in a code generation task to assess metacognitive reasoning while integrating either human programmers or additional reasoning models.

2.3 Modeling Uncertainty

Writing quality code can be difficult for both humans and AI, partially due to the challenge of when to spend more temporal and reasoning resources and when to go with one's first instinct. Much of the time spent writing code is so-called *boilerplate* or code that humans have written many times in the past, only requiring small changes. However, important parts of programming decisions require more thought when writing, with significant negative impacts to code quality arising from incorrect solutions at critical points [35, 21, 40]. This is an important aspect of metacognitive reasoning, since a high degree of awareness over uncertainty allows humans to allocate more computational and temporal resources to these critical points in decision making. In EBTs this issue is addressed by training a model (Algorithm 1) to assign a scalar energy value to inputs which corresponds to their likelihood [20]. EBTs leverage probabilistic Energy-Based Models (EBMs) that define a probability distribution using a Botlsmann distribution $p_\theta(x) = \exp(-E_\theta(x))/Z(\theta)$ where $Z(\theta) = \int \exp(-E_\theta(x))dx$ [20, 9, 13]. This allows for a deeper understanding of when the model is more or less sure about its output. Though importantly, this probability distribution is not the typically LLM formulation of uncertainty, as the probabilities do not require normalization and energies are measured in arbitrary units making direct comparison of different model energy predictions difficult [26]

2.4 Prediction Verification

EBTs allow for prediction verification by using the forward passes of the model at inference time to effectively replicate a GAN discriminator by producing an energy verification [20]. Verifying predictions is another important quality of agents with metacognition abilities [46]. When humans attempt to verify their reasoning, they typically try to think about their answers using a different reasoning method than the one that they used to generate their answers [22]. This allows humans to determine the accuracy of their answers by confirming them in multiple different ways, while also allowing them to flexibly reason about their previous behaviors and thought processes [10]. In our proposed assessment pipeline, prediction verification is not only a method of generating next token predictions but also a way for the model to reassess previously generated output based on feedback from external sources.

2.5 Dynamic Compute Allocation

EBTs can control the number of energy optimization steps that they use at inference to find the best solution [20]. In real-world problems, there is often a challenge of how much compute resources to allocate to finding a solution, since spending too many resources on problems that are simple or less impactful can take up resources that would be better allocated elsewhere [27, 31]. We propose that this dynamic compute allocation feature of EBTs is highly related to metacognitive reasoning. As a result, EBTs can think harder about their responses depending on a variety of features of the task at hand. For instance, humans can respond to both low certainty and high impact of decisions by thinking harder about a problem [19]: which they can control through metacognition. When humans are less confident in their answers or they know their answers are highly impactful, they can allocate more cognitive resources to different areas of a decision making problem [4].

3 Metacognition Assessment Pipeline

Our proposed approach to assessing the metacognitive reasoning in code generation of Energy-Based Transformers is shown in Figure 1 using two examples of possible model output, one of the correct next token prediction and the other of the incorrect next token prediction. Importantly, in this example the output is a function call '.iterrows()' but in reality would be a single token. This illustrative example is shown to describe how our proposed method could be used to generate code in a collaborative and agentic context. In both the correct and incorrect next token prediction examples,

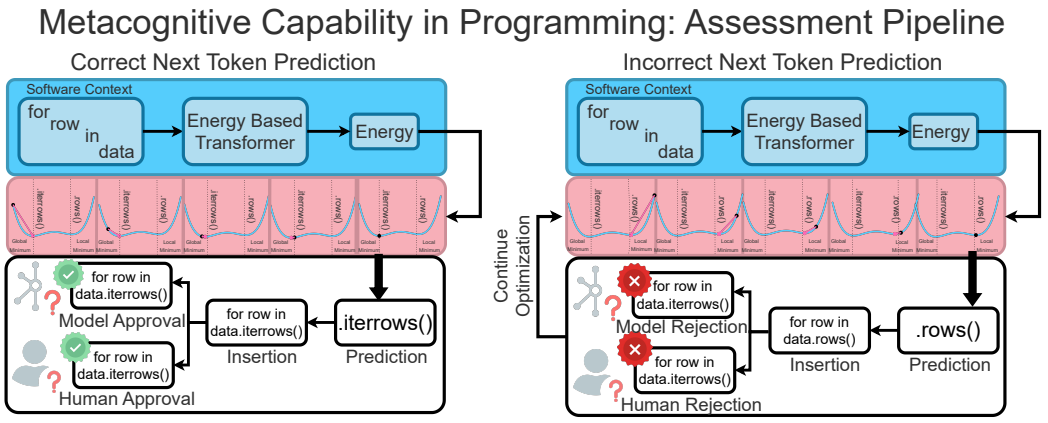

Figure 1: Diagram of our proposed assessment pipeline for testing metacognitive reasoning capability in code generation. **Top:** Both blue regions show next token prediction for software generation in which the context is the existing generated code which is input into the EBT to make a prediction of the energy landscape. **Middle:** In pink, this energy is minimized through multiple forward passes to find the lowest energy and highest probability next token. **Left White**: In this correct next token prediction example, the energy optimization process began near the global minimum of the correct response '.iterrows()'. This leads to the correct next token which is inserted into the code before being approved by either a human programmer or a separate LLM model. **Right White**: In this incorrect next token prediction example, the energy optimization begins near a local minimum of the incorrect but plausible sounding next token '.rows()'. This leads to the incorrect next token prediction which is inserted into the code before being rejected by either the human programmer or separate LLM model. After rejection, the optimization step is repeated in an attempt to find the true global minimum.

the code context is the same 'for row in data', but the location of the initial predicted token within the energy optimization step is different. In the correct next token prediction example, the correct next token '.itterows()' corresponds to both the local and global minimum, meaning that the process of repeated forward passes is able to find the correct solution.

In the incorrect next token prediction example, there is a local minimum '.rows' that corresponds to an incorrect but reasonably likely next token to predict. As a result of this local minimum, the forward passing of the model is not able to find the correct next token to predict. Here, we suggest an intervention using human approval which, when denied, will loop back to the energy optimization step and produce additional model forward passes in an attempt to find the global minimum and thus the correct response. In this assessment, updating the next token prediction to the correct token after receiving feedback represents correct reasoning, while failure represents incorrect reasoning.

Algorithm 3 describes our approach to integrating a human into the pipeline of metacognition assessment. This is done by replicating the inference time algorithm of the standard EBT approach and augmenting it with a human programmer. This human programmer is assumed to take the form of a function $H(\cdot)$ that takes as input the output of the model $\hat{y}^*$ and categorizes it as either correct $H(\hat{y}^*) = 1$ or incorrect $H(\hat{y}^*) = 0$. After the output $\hat{y}^*$ is generated, the human determines if it is correct, if it isn't $H(\hat{y}^*) \neq 1$ then the number of steps is increased by the step increase hyperparameter $N \leftarrow N + n_1$. If it is correct then the output is accepted. We only perform one check for the accuracy of the output and use a binary categorization because we assume that human programmers may not be able to give highly accurate measure of confidence for their categorization, and that querying them for accuracy categorization will be costly.

The method for integrating a similar approach as the human in the loop method but instead using an LLM to categorize code accuracy is described in Algorithm 4. Here, instead of only categorizing output as correct or incorrect we assume a range of possible confidence scores for the output of the main code generation method. In this example, we show two different thresholds t_0, t_1 for code generation confidence. This allows us to use the same number of separate step increase hyper parameters n_0, n_1 to increase the number of steps for the energy minimization process, depending on the confidence threshold. In application a range of possible sizes of thresholds and hyper parameters

is possible. This is done with the assumption that LLM categorization will be less costly than human categorization, but still costly enough to necessitate avoiding categorization at every step of the energy minimization process.

Algorithm 3: HUMAN IN THE LOOP METACOGNITION ASSESSMENT

Input: Context x, EBM $E_\theta(x, \hat{y})$ Human $H(\cdot)$
Hparams: Steps N, Step Size α, Samples M, Step Increase n

for $j = 1, \ldots, M$ **do**
  Sample $\hat{y}_{0,j} \sim \mathcal{N}(0, I)$;
  for $i = 0, \ldots, N-1$ **do**
    $\hat{y}_{i+1,j} \leftarrow \hat{y}_{i,j} - \alpha \nabla_{\hat{y}_{i,j}} E_\theta(x, \hat{y}_{i,j})$;
return $\hat{y}^* = \text{argmin}_j \, E_\theta(x, \hat{y}_{N,j})$;
if $H(\hat{y}^*) \neq 1$ **then**
  $N \leftarrow N + n$
  go to: 1

Algorithm 4: LLM ASSISTED METACOGNITION ASSESSMENT

Input: Context x, EBM $E_\theta(x, \hat{y})$ LLM $M(\cdot)$
Hparams: Steps N, Step Size α, Samples M Step Increases n_0, n_1, Thresholds t_0, t_1

for $j = 1, \ldots, M$ **do**
  Sample $\hat{y}_{0,j} \sim \mathcal{N}(0, I)$;
  for $i = 0, \ldots, N-1$ **do**
    $\hat{y}_{i+1,j} \leftarrow \hat{y}_{i,j} - \alpha \nabla_{\hat{y}_{i,j}} E_\theta(x, \hat{y}_{i,j})$;
return $\hat{y}^* = \text{argmin}_j \, E_\theta(x, \hat{y}_{N,j})$;
if $M(y) < t_0$ **then**
  $N \leftarrow N + n_0$
  go to: 1
else if $M(\hat{y}^*) < t_1$ **then**
  $N \leftarrow N + n_1$
  go to: 1

4 Discussion

In this work, we propose a method for assessing the metacognitive reasoning capabilities of EBTs using two different possible methods. In the first, a human programmer is used to verify or reject possible responses that EBTs generate. In the second, an LLM is used to assess the accuracy of responses, either the same model or a separate one trained specifically to verify code. In both of these methods, 'passing' the assessment corresponds to correctly identifying the next token to generate after recieving feedback. Alongside these proposed assessment methods we describe two algorithms that slightly edit the inference time algorithm of EBTs, to account for this external feedback.

The three central capabilities of EBTs that we are interested in assessing with respect to metacognitive reasoning capability are predicting uncertainty, validating their predictions, and allowing for dynamic allocation of compute resources. EBTs are unique from other LLM methods in that these three features are inherent to their functioning, rather than estimated or calculated separately as an external functionality. This is important as it means that other LLM approaches do not allow for a better understanding of where and why there may be issues in generated code. This fits in well with human-in-the-loop applications because it can direct programmers attention to areas of the code where the EBT model is potentially more fragile, where energy was not spent, and help users understand why models made mistakes, at least in relation to energy.

There are some limitations to the specific method of assessing metacognition that we describe in this work, though we believe that through simple alterations these can be accounted for. One issue to using the LLMs $M(\cdot)$ and Humans $H(\cdot)$ to check the code generated by the main model is that this would significantly slow down the LLM generated coding process. However, we believe that this issue can be accounted for by only querying the human or LLM to check the response on occasion. This could be controlled through the introduction of an additional hyperparameter that controls when to check with the Human or LLM based on the final energy at the last step of energy optimization $E_\theta(x, \hat{y}_{N,j})$. Another option would be to generate full code blocks, with additional information on the amount of energy used on each token, to draw programmer attention to specific areas.

References

[1] Rakefet Ackerman and Valerie A Thompson. "Meta-reasoning: Monitoring and control of thinking and reasoning". In: *Trends in cognitive sciences* 21.8 (2017), pp. 607–617.

[2] Ahmet Oguz Akturk and Ismail Sahin. "Literature review on metacognition and its measurement". In: *Procedia-Social and Behavioral Sciences* 15 (2011), pp. 3731–3736.

[3] Jaan Aru, Matthew E Larkum, and James M Shine. "The feasibility of artificial consciousness through the lens of neuroscience". In: *Trends in neurosciences* 46.12 (2023), pp. 1008–1017.

[4] Christopher J Bates et al. "Adaptive allocation of human visual working memory capacity during statistical and categorical learning". In: *Journal of vision* 19.2 (2019), pp. 11–11.

[5] Emily M Bender et al. "On the dangers of stochastic parrots: Can language models be too big?" In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021, pp. 610–623.

[6] Sébastien Bubeck et al. "Sparks of artificial general intelligence: Early experiments with gpt-4". In: *arXiv preprint arXiv:2303.12712* (2023).

[7] Andy Clark. "Whatever next? Predictive brains, situated agents, and the future of cognitive science". In: *Behavioral and brain sciences* 36.3 (2013), pp. 181–204.

[8] Brendan Conway-Smith and Robert L West. "System-1 and System-2 realized within the Common Model of Cognition". In: *AAAI 2022 fall symposium*. 2022.

[9] Anna Dawid and Yann LeCun. "Introduction to latent variable energy-based models: a path toward autonomous machine intelligence". In: *Journal of Statistical Mechanics: Theory and Experiment* 2024.10 (2024), p. 104011.

[10] Annemie Desoete. "Multi-method assessment of metacognitive skills in elementary school children: How you test is what you get". In: *Metacognition and learning* 3.3 (2008), pp. 189–206.

[11] Aniket Didolkar et al. "Metacognitive Capabilities of LLMs: An Exploration in Mathematical Problem Solving". In: *arXiv preprint arXiv:2405.12205* (2024).

[12] Aniket Didolkar et al. "Metacognitive reuse: Turning recurring llm reasoning into concise behaviors". In: *arXiv preprint arXiv:2509.13237* (2025).

[13] Yilun Du and Igor Mordatch. "Implicit generation and modeling with energy based models". In: *Advances in neural information processing systems* 32 (2019).

[14] Jonathan St BT Evans. "In two minds: dual-process accounts of reasoning". In: *Trends in cognitive sciences* 7.10 (2003), pp. 454–459.

[15] Jonathan St BT Evans and Keith E Stanovich. "Dual-process theories of higher cognition: Advancing the debate". In: *Perspectives on psychological science* 8.3 (2013), pp. 223–241.

[16] Pablo Fernández Velasco and Slawa Loev. "Metacognitive feelings: A predictive-processing perspective". In: *Perspectives on Psychological Science* 20.4 (2025), pp. 691–713.

[17] Stephen M Fleming and Hakwan C Lau. "How to measure metacognition". In: *Frontiers in human neuroscience* 8 (2014), p. 443.

[18] Logan Fletcher and Peter Carruthers. "Metacognition and Reasoning". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 367.1599 (2012), pp. 1366–1378. DOI: 10.1098/rstb.2011.0413.

[19] Gerd Gigerenzer. *Rationality for mortals: How people cope with uncertainty*. Oxford University Press, 2010.

[20] Alexi Gladstone et al. "Energy-Based Transformers are Scalable Learners and Thinkers". In: *arXiv preprint arXiv:2507.02092* (2025).

[21] Abram Hindle et al. "On the naturalness of software". In: *Communications of the ACM* 59.5 (2016), pp. 122–131.

[22] Philip N Johnson-Laird. "Mental models and human reasoning". In: *Proceedings of the National Academy of Sciences* 107.43 (2010), pp. 18243–18250.

[23] Daniel Kahneman. "Thinking, fast and slow". In: *Farrar, Straus and Giroux* (2011).

[24] Daniel Kahneman, Shane Frederick, et al. "Representativeness revisited: Attribute substitution in intuitive judgment". In: *Heuristics and biases: The psychology of intuitive judgment* 49.49-81 (2002), p. 74.

[25] Tamera Lanham et al. "Measuring faithfulness in chain-of-thought reasoning". In: *arXiv preprint arXiv:2307.13702* (2023).

[26] Yann LeCun et al. “A tutorial on energy-based learning”. In: *Predicting structured data* 1.0 (2006).

[27] Falk Lieder and Thomas L Griffiths. “Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources”. In: *Behavioral and brain sciences* 43 (2020), e1.

[28] Andrew Lizarraga, Edouardo Honig, and Ying Nian Wu. “From Stochastic Parrots to Digital Intelligence: The Evolution of Language Models and Their Cognitive Capabilities”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 17.3 (2025), e70035.

[29] Qing Lyu et al. “Faithful chain-of-thought reasoning”. In: *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*. 2023.

[30] Aman Madaan et al. “Self-Refine: Iterative Refinement with Self-Feedback”. In: *arXiv preprint arXiv:2303.17651* (2023).

[31] Tailia Malloy and Chris R. Sims. “Efficient Visual Representations in Learning and Decision Making”. In: *Psychological Review* (2024).

[32] Tailia Malloy et al. “Improving Online Anti-Phishing Training Using Cognitive Large Language Models”. In: *Computers and Human Behavior* (2025).

[33] James A Marcum. “An integrated model of clinical reasoning: dual-process theory of cognition and metacognition”. In: *Journal of evaluation in clinical practice* 18.5 (2012), pp. 954–961.

[34] Thomas Metzinger. “Artificial suffering: An argument for a global moratorium on synthetic phenomenology”. In: *Journal of Artificial Intelligence and Consciousness* 8.01 (2021).

[35] Thomas J Ostrand, Elaine J Weyuker, and Robert M Bell. “Predicting the location and number of faults in large software systems”. In: *IEEE Transactions on Software Engineering* 31.4 (2005), pp. 340–355.

[36] Rafael Rafailov et al. “Direct preference optimization: Your language model is secretly a reward model”. In: *Advances in neural information processing systems* 36 (2023).

[37] Dobromir Rahnev. “A comprehensive assessment of current methods for measuring metacognition”. In: *Nature Communications* 16.1 (2025), p. 701.

[38] Matthew G Rhodes. “Metacognition”. In: *Teaching of Psychology* 46.2 (2019), pp. 168–175.

[39] Max Rollwage and Stephen M Fleming. “Confirmation bias is adaptive when coupled with efficient metacognition”. In: *Philosophical Transactions of the Royal Society B* 376.1822 (2021), p. 20200131.

[40] Chanchal Kumar Roy and James R Cordy. “A survey on software clone detection research”. In: *Queen’s School of computing TR* 541.115 (2007), pp. 64–68.

[41] Nicholas Shea et al. “Supra-personal cognitive control and metacognition”. In: *Trends in cognitive sciences* 18.4 (2014), pp. 186–193.

[42] Valerie A Thompson, JSBT Evans, and K Frankish. “Dual process theories: A metacognitive perspective”. In: *Ariel* 137.51 (2009), p. 43.

[43] Haoye Tian et al. “Is ChatGPT the ultimate programming assistant–how far is it?” In: *arXiv preprint arXiv:2304.11938* (2023).

[44] Shengbang Tong et al. “Eyes wide shut? exploring the visual shortcomings of multimodal llms”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 9568–9578.

[45] Rodolfo Valiente and Praveen K Pilly. “Metacognition for Unknown Situations and Environments (MUSE)”. In: *Neural Networks* (2025), p. 108131.

[46] Katrin Vorhölter. “Conceptualization and measuring of metacognitive modelling competencies: Empirical verification of theoretical assumptions”. In: *Zdm* 50.1 (2018), pp. 343–354.

[47] Xuezhi Wang et al. “Self-consistency improves chain of thought reasoning in language models”. In: *arXiv preprint arXiv:2203.11171* (2022).

[48] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[49] Colin White et al. “LiveBench: A Challenging, Contamination-Free LLM Benchmark”. In: *The Thirteenth International Conference on Learning Representations*. 2025.

[50] Yujia Zhou et al. “Metacognitive Retrieval-Augmented Large Language Models”. In: *arXiv preprint arXiv:2402.11626* (2024).