Supplementary Material for Thought Cloning: Learning to Think while Acting by Imitating Human Thinking

Anonymous Author(s)
Affiliation
Address
email

A Architecture and Training Details

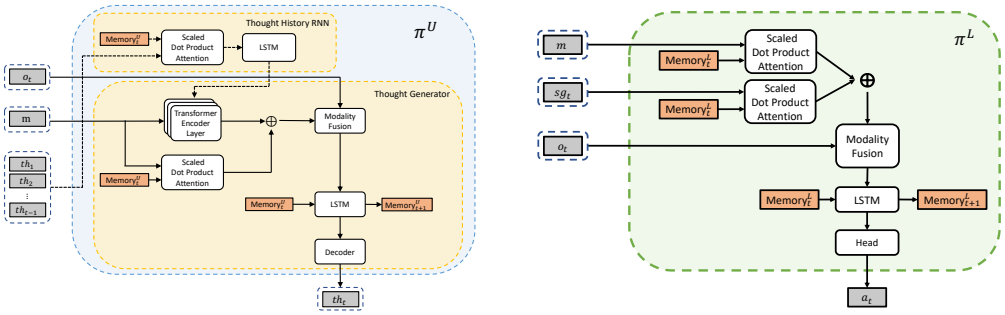

Figure 1: Detailed architecture for Thought Cloning (TC) agent. At each timestep t, the inputs to the TC agent include a natural language-defined mission m, an observation o_t, and all preceding thoughts $\{th_\tau\}_{\tau=1}^{t-1}$. The generated thought th_t from the Upper-level Component will be the input to the Lower-level Component and an action a_t is predicted by the Lower-level Component. (**Left**): The Upper-level Component. We employ an LSTM [1] to embed the thought history and a transformer encoder to process both the mission and thought history. The text input is then fused with the visual observation input using FiLM [2]. (**Right**): The Lower-level Component is largely similar to the BabyAI agent [3], with the primary difference being the additional embedding of the thought generated by the Upper-Level Component.

Table 1: Hyperparameter Settings

Hyperparameter	**Value**
Adam β_1	0.9
Adam β_2	0.99
Adam ϵ	10^{-5}
Entropy Coefficient	0.01
Image Embedding Dimension	128
Text Embedding Dimension	256
Memory Dimension	2048

For full transparency, replicability, and to facilitate future research building on our work, we are releasing both the source code and model weights. Additionally, we provide key details necessary for the evaluation and replication of our work in this supplementary information. The architectural details

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Algorithm 1 Thought Cloning

1: **Input:** thought dataset $\mathcal{D} = \{D_i\}_{i=1}^{N}$, where each $D_i = (m, \{(o_t, th_t, a_t)\}_{t=1}^{T})$, upper level component $\pi_{\theta_u}(th|o, m, \{history\_th\})$, lower level component $\pi_{\theta_l}(a|o, m, th)$
2: **while** training **do**
3: **for** each $D_i = (m, \{(o_t, th_t, a_t)\}_{t=1}^{T})$ in $\mathcal{D}$ **do**
4: **for** each (o_t, th_t, a_t) in D_i **do**
5: Generate upper level thought sentence $\hat{th}_t = \pi_{\theta_u}(\cdot|o_t, m, \{th_\tau\}_{\tau=1}^{t-1})$
6: Predict lower level action probability distribution $\hat{a}_t = \pi_{\theta_l}(\cdot|o_t, m, \hat{th}_t)$
7: Compute the loss: $\mathcal{L}(\theta_u, \theta_l) = \mathcal{L}_{CE}(a_t, \hat{a}_t) + \alpha\mathcal{L}_{CE}(th_t, \hat{th}_t) - \beta H(\hat{a}_t)$
8: Update the policy network parameters θ_u, θ_l by minimizing $\mathcal{L}(\theta_u, \theta_l)$
9: **end for**
10: **end for**
11: **end while**

of Thought Cloning models are shown in Fig. 1. As in [4], all missions and thoughts are encoded with Gated Linear Units (GLUs), with separate encoders employed for the missions and thoughts respectively. After the encoding process, we apply an attention mechanism [5] to dynamically weight the importance of different parts of the text encoding, based on the state history. The observation is encoded with a Convolutional Neural Network (CNN) and a Bag-of-Words [6] encoding approach. In the Upper-level Component, a Transformer encoder [5, 7] is adopted to embed the thought history and mission, with the thought history as the query and the mission as the key and value. This Transformer encoder consists of two layers, each with two heads. The Lower-level Component is identical to the Behavior Cloning Baseline, except with the additional encoding of thoughts. Key architectural parameters, such as memory size and embedding size, are consistent with the baseline in [4], as shown in Table 1.

The pseudocode for Thought Cloning (TC) training framework is shown in Algorithm 1. In the loss function, we follow [4] by including an entropy term for actions. The Adam optimizer [8] is adopted to train TC and TC variant, with a batch size of 180 and a learning rate of $5e^{-4}$. Similar to the setting in baseline [4, 3], we train BC with a batch size of 296 and a learning rate $5e^{-5}$. The learning rate schedule begins with a warm-up phase of 5 training steps, linearly increasing from $1e^{-4}$ to $5e^{-4}$, and then decaying by 50% at 120th training steps, similar to the practices in [3, 9]. In line 5 of Algorithm 1, the input thought could be the ground truth from the dataset (th_t) or the generated thought from the Upper-level Component ($\hat{th}_t$), depending on with or without teacher forcing. For training efficiency, Backpropagation Through Time was truncated at 20 steps in TC. The mix precision in PyTorch is also adopted during training, which speeds up training without sacrificing much performance [10]. In fine-tuning experiments, due to the increased difficulty of the levels and longer steps requiring more memory, we reduced the batch size from 180 to 40 and trained with an auto-regressive strategy. Detailed hyperparameter settings are shown in Table 1.

B Synthetic Human Thought Dataset

Fig. 2 presents an example trajectory. We translate the inner state of the BabyAI Oracle Solver (called "Bot" in [3]) into natural language thoughts. These thoughts outline the current plan for task completion and also describe the underlying intentions behind these plans, as the same low-level plan can serve different stated high-level purposes. For instance, the plan could be to *"open the red door"* with the intention of *"completing the open mission"* or *"exploring"*. The segments with inserted noise are marked in red in Fig. 2.

C Example on Diagnosing Agents by Observing Thoughts

In this section, we provide an example of one time when we were able to diagnose Thought Cloning (TC) agents by observing their thoughts during the development phase of this paper. In the early stages of development, we trained the TC agent with a constant teacher-forcing strategy. We observed that during testing, the agents often got stuck persisting with incorrect thoughts and did not actively explore new ideas. For instance, in the top right example in Fig. 3, after t=53, the agent persistently

mission: put a key next to the blue ball and pick up the green key

t=1-5: Thought "pickup a key to complete PutNext mission", Action [forward, right, ... , forward]
t=6: Thought "pickup blue key to complete PutNext mission", Action [pickup]
t=7-12: Thought "explore unseen area", Action [right, right, ... , right]
t=13-14: Thought "open green door to explore", Action [left, left, ... , toggle]
t=15-22: Thought "explore unseen area", Action [forward, forward, ... , left]
t=23-28: Thought "open red door to explore", Action [forward, forward, ... , toggle]
t=29: Thought "explore unseen area", Action [forward]
t=30-38: Thought "drop blue key to complete PutNext mission", Action [forward, forward, ... , drop]
t=39-44: Thought "explore unseen area", Action [right, right, ... , left]
t=45-48: Thought "open green door to explore", Action [left, left, ... , toggle]
t=49-56: Thought "explore unseen area", Action [forward, forward, ... , left]
t=57-69: Thought "open green door to explore", Action [forward, forward, ... , toggle]
t=70-77: Thought "explore unseen area", Action [forward, forward, ... , right]
t=78-88: Thought "open grey door to explore", Action [forward, forward, ... , toggle]
t=89-95: Thought "explore unseen area", Action [forward, forward, ... , right]
t=96-101: Thought "open purple door to explore", Action [forward, forward, ... , toggle]
t=102-110: Thought "explore unseen area", Action [forward, forward, ... , right]
t=111-122: Thought "open red door to explore", Action [forward, forward, ... , toggle]
t=123-130: Thought "explore unseen area", Action [forward, forward, ... , right]
t=131-134: Thought "open grey door to explore", Action [right, right, ... , forward]
t=135-137: Thought "pickup green box to complete PutNext mission", Action [left, left, left]
t=138-204: Thought "open grey door to explore", Action [right, right, ... , toggle]
t=205: Thought "explore unseen area", Action [forward]
t=206-213: Thought "pickup green key to complete pickup mission", Action [forward,...,pickup]

Figure 2: Example trajectories of the synthetic human thought dataset. The inserted noisy segments are highlighted in red.

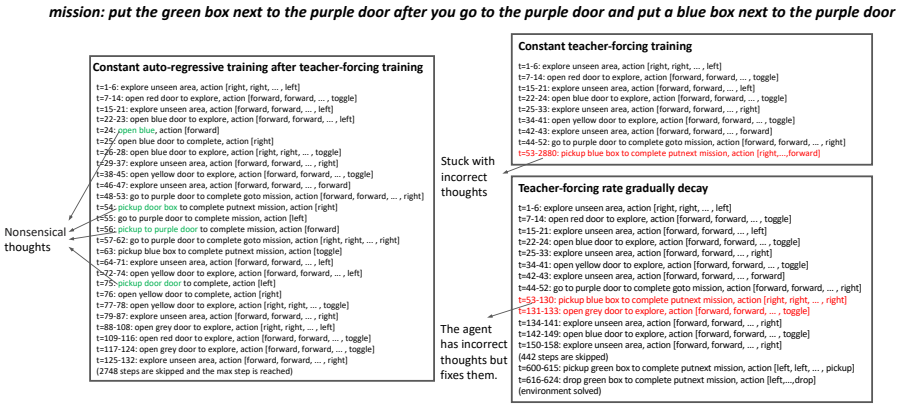

Figure 3: Example trajectories of agents trained with different strategies. **Constant teacher-forcing training** refers to exclusively training with the teacher-forcing strategy. In this scenario, the agent does not learn to recover from incorrect thoughts. Once it adopts an incorrect thought, it continues to follow this thought for thousands of time-steps until it reaches the maximum step count (top right from t=53 to t=2880). **Constant auto-regressive training after teacher-forcing training** implies directly transitioning to auto-regressive training following an initial phase of teacher-forcing training. In this case, agents begin to generate nonsensical thoughts, as shown on the left, such as *open blue* at t=24 (left) and *pickup door door* at t=75 (left). **Gradual decay of teacher-forcing rate** involves gradually reducing the ratio of teacher-forcing during training. This strategy is adopted in the final version of Thought Cloning. In this setting, the agent might generate some incorrect thoughts as shown at t=53 (bottom right), but it can recover from these errors to explore new ideas, as evidenced at t=131 (bottom right).

attempted to implement the incorrect thought *"pickup blue box to complete putnext mission"* until it reached the maximum step limit, without seeking new ideas. This observation led us to realize that, as we exclusively trained the agent with oracle thoughts via a teacher-forcing strategy, the agent had never practiced dealing with incorrect thoughts and consequently had not learned to recover from them by trying alternative ideas. Subsequently before this realization, we had attempted to transition directly to auto-regressive training after the teacher-forcing training stage. However, the agent then started to generate nonsensical thoughts. The trajectory in Fig. 3 (left) shows nonsensical thoughts such as *open blue* (t=24) and *pickup door door* (t=75) being generated when a constant auto-regressive strategy is applied. Because of the realization from being able to observe the agent's thoughts, we adopted a gradual decay schedule for teacher-forcing rates during training. As shown in Fig. 3 (bottom right), the agent in this setting was able to explore new ideas after failing on an incorrect thought, and it rarely generate nonsensical thoughts. For example, the agent generates an incorrect thought at t=53, but it can recover from these errors to explore new ideas, e.g. *open grey door to explore*. Because we can observe the TC agents *thinking out loud*, we are able to identify the issue and improve the agent's performance. Without this visibility into the agent's thoughts, simply observing their actions would have made it much harder to pinpoint the underlying problems.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[2] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[3] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *International Conference on Learning Representations*, 2019.

[4] David Yu-Tung Hui, Maxime Chevalier-Boisvert, Dzmitry Bahdanau, and Yoshua Bengio. Babyai 1.1. *arXiv preprint arXiv:2007.12770*, 2020.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[7] Gi-Cheon Kang, Jaeseo Lim, and Byoung-Tak Zhang. Dual attention networks for visual reference resolution in visual dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 2024–2033, 2019.

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[10] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.