Practice Makes Fluent: Learning to Use Earlier Predictions in Looped Models

Minseok Seo† Hee-Seon Kim† Changick Kim

School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Republic of Korea

minseok.seo@kaist.ac.kr hskim98@kaist.ac.kr changick@kaist.ac.kr

† These authors contributed equally.

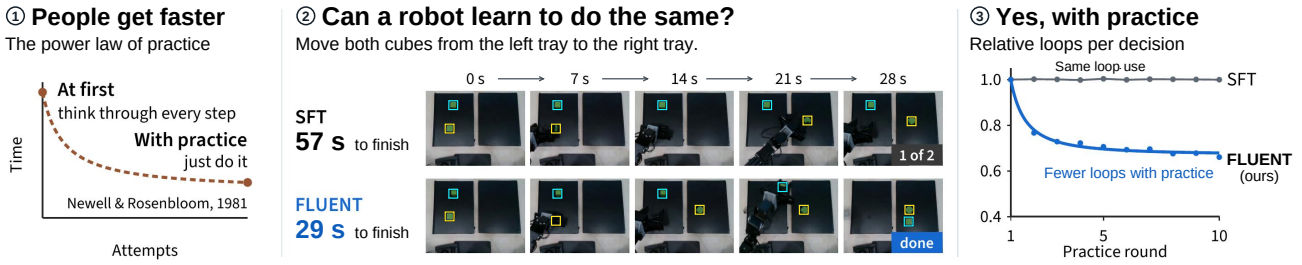

Figure 1. **Practice changes which predictions a looped policy can use.** The center shows end-to-end real-robot pick-and-place: successful episodes take 31 s on average for FLUENT and 69 s for SFT (illustrated: 29 s and 57 s), with fewer chunks and lower per-chunk time.

Abstract

People become faster at a task through practice, but models that improve with experience do not necessarily learn to use less computation. We train looped models, whose intermediate predictions allow earlier exits, to perform tasks with fewer iterations through practice. This is difficult because shared-weight updates can degrade the final-loop prediction while improving early ones, and the exit gate must track early predictions as their quality changes. We propose FLUENT, which requires no additional task labels or rewards: it distills the initial model's final-loop prediction into early predictions, anchors the current final-loop prediction to the same frozen teacher, and trains the exit gate to balance disagreement with the current final-loop prediction against computation cost. FLUENT improves performance at shallower depth in new Crafter worlds, raises LIBERO success with fewer action-head iterations, and maintains language-model accuracy at lower selected depth. On a real robot, success rises from 62.3% to 74.6% while mean time over successful episodes falls from 69 s to 31 s.

1. Introduction

A person who repeats a task gets faster at it. Execution time often falls steeply over the first attempts and more gradually thereafter [29, 31] (Fig. 1, left). Robots and agents also perform similar tasks repeatedly, from visual manipulation [3, 4, 21, 42] to language-guided interaction [39, 43]. Yet better task performance through experience does not necessarily mean less computation at inference.

Early exit and adaptive-depth methods allocate computation according to the current input [28, 37, 42, 47, 50, 55]. Our question is complementary: **can experience make a previously difficult task require less computation?** Methods that reuse reasoning traces [6, 33, 39], skills [32, 43], or cached computation [7, 13, 49] exploit previous experience through external records. These records must be stored, retrieved, and matched to the current task. We instead store experience in the model's weights: the model learns through self-distillation from the task inputs it encounters, without additional labels or rewards. We call this process *practice*, and aim to improve task performance while reducing the computation needed to perform the task.

We study this question in looped models [19, 42, 55]. In the architectures we consider, a shared block repeatedly updates a latent state, and a shared output head can read a

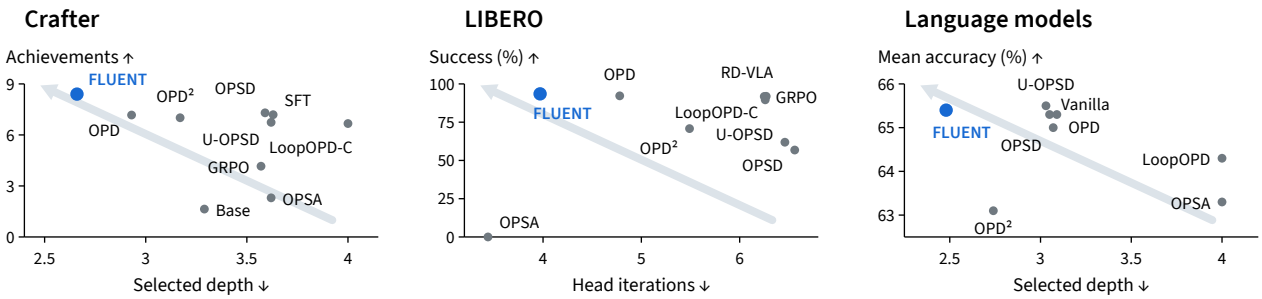

Figure 2. **Performance against depth in three benchmark settings.** Points use Tables 1, 2, and 4; upper left is better. Crafter and language models use selected output depth, whereas LIBERO uses executed action-head iterations. Arrows indicate the preferred direction.

prediction after each loop. This makes earlier predictions available, but availability alone does not make them useful. Practice must both improve these predictions and enable the policy to act on them. A natural starting point is to use the frozen initial model as a teacher, distilling its deep predictions into shallow loops.

However, this addresses only part of this problem. It directly supervises agreement between the first loop and the teacher, not the task performance of every subsequent loop. Because the weights are shared, later predictions also change, but they need not improve together. Moreover, even an improved first-loop prediction saves computation only when the exit gate selects it. Improving early predictions and learning to use them are distinct aspects of practice.

We present FLUENT, a self-distillation procedure that addresses these two parts together. Translate trains early predictions to match the final-loop output of the frozen initial model. Because these updates also affect later loops, Anchor keeps the current final-loop prediction close to the same frozen teacher. This anchored prediction provides the reference for training the gate. Gate adapts the exit policy to the updated predictions, balancing agreement with the current final loop against the cost of further computation. Practice uses no additional task labels or rewards, and inference requires no external experience store.

We evaluate FLUENT in four settings, examining both prediction depth and task performance. In Crafter [14], FLUENT improves performance in new worlds while selecting shallower predictions. On LIBERO [26], FLUENT executes 36.7% fewer action-head iterations than RD-VLA [42], while improving success. On a real robot, relative to the SFT policy, FLUENT increases success from 62.3% to 74.6% and reduces mean completion time over successful episodes from 69 s to 31 s. In language-model distillation, compared with the vanilla model using its original gate, FLUENT maintains mean accuracy while reducing mean selected depth by 18.7%.

2. Related Work

Looped models and adaptive depth. Adaptive inference allocates computation across inputs or exits from intermediate predictions [10, 28, 37, 40, 47]. Ouro learns predictions and an exit distribution across recurrent depths [55]; LoopFormer aligns short and long paths [19]. In manipulation, RD-VLA refines an action head recurrently [42], while other methods use intermediate actions or learned stopping [18, 22, 50]. Policy distillation can also reduce a fixed inference depth [20]. We study changes in both predictions and exits during practice. Tokenwise early termination requires a cache policy such as Ragged KV [30].

Reusing experience. Experience-dependent reasoning cost has been studied with memory-based systems [34]. Agents store traces, feedback, and executable skills for later reuse [32, 33, 39, 43, 46, 48]. LAG represents past reasoning logs as KV states, and FCGraft reuses functions and their caches to accelerate code-policy generation [6, 7]. Other cache systems reduce repeated prompt processing [13, 49, 54]. Our setting updates policy parameters rather than retrieving prior episodes or cached task solutions at inference; it is complementary to retrieval.

On-policy and self-distillation. On-policy distillation trains on contexts generated by the student [1, 17]. OPSD and U-OPSD construct supervision from the model's own predictions under different contexts [23, 53]; OPSA instead adapts using entropy-derived advantages without a teacher [11]. These methods do not explicitly optimize recurrent exit depth, even when they improve other efficiency measures. Self-distilled multi-exit networks also teach intermediate predictions [28, 35, 52], and distillation can preserve a prior model's outputs during further learning [24]. We use these principles jointly in a recurrent policy whose shallow and deep predictions share the same block.

Distillation across loops. LoopOPD distills a frozen four-loop teacher into a two-loop student, and D-LoopOPD uses the current four-loop prediction as the teacher [45]; both report gains at intermediate and final depths but compare methods on four-loop outputs and learn no exit policy. Such

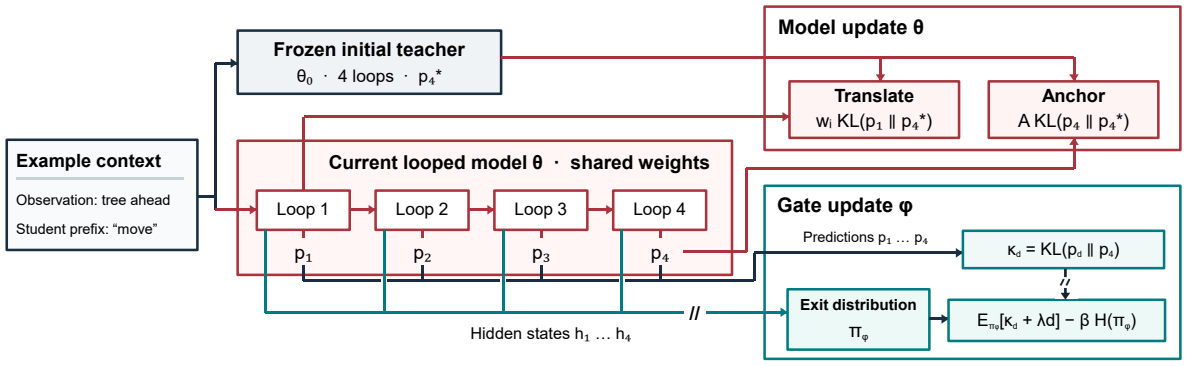

Figure 3. **Overview of FLUENT.** The initial model's final prediction supervises early predictions (Translate) and the student's final prediction (Anchor). The gate learns a trade-off between disagreement with the current final prediction and depth (Gate). Deep discrepancies are training targets, not inference-time inputs to the gate. Double slashes denote stopped gradients.

gains are not guaranteed under shared-weight updates, motivating our explicit preservation of the final-loop prediction. LoOPSD distills a teacher built from searched layer-repetition routes into a standard-path student [12]. A better shallow prediction yields an earlier output only if the exit policy selects it, and shared-weight updates shift the final prediction the gate is trained against. FLUENT therefore pairs Anchor, which keeps the final prediction aligned with the frozen initial teacher, with Gate, which balances prediction discrepancy and computation cost.

3. Method

3.1. Preliminaries: looped models and early exit

For context c, a shared block with parameters θ produces predictions $p_d^\theta(\cdot \mid c)$ at depths $d = 1, \ldots, D$. We restrict attention to architectures with a depth-wise readout through a shared output module. The token gate has parameters ϕ and stopping hazards based on the hidden states available up to each depth. These hazards define an exit distribution $\pi_\phi(d \mid c)$. For threshold $q \in (0, 1]$, we select

$$e(c) = \min\left\{d : \textstyle\sum_{j=1}^{d} \pi_\phi(j \mid c) \geq q\right\}. \tag{1}$$

Lower $e(c)$ means an earlier output.

3.2. Practice: translating deep predictions to a shallow loop

Let θ_0 denote the model parameters before practice. We initialize the student at θ_0 and keep the initial model frozen, using its final-loop prediction as the teacher: $p_D^\star(\cdot \mid c) = p_D^{\theta_0}(\cdot \mid c)$. In interactive settings, practice proceeds in rounds: the agent interacts with the environment using its current policy and then trains on the states it visited.

On each visited state, the shallow student ($d = 1$) generates a response. For each token context $c_1, \ldots, c_N$ in that response, Translate matches the shallow prediction to the frozen teacher:

$$\mathcal{L}_{\mathrm{T}} = \frac{1}{N} \sum_{i=1}^{N} w_i \, \mathrm{KL}\big(p_1^\theta(\cdot \mid c_i) \,\big\|\, p_D^\star(\cdot \mid c_i)\big)\,. \tag{2}$$

The weight w_i is proportional to the highest token probability of the teacher, placing more weight on contexts where the teacher is confident. Because the student generates the response itself, distillation uses token contexts encountered by the shallow student rather than teacher-generated contexts. The frozen model supervises practice, so no additional rewards, labels, or demonstrations are required.

3.3. Anchor: holding the final loop in place

Shared parameters couple the shallow and final predictions. An update to the shallow prediction can therefore change the final output, even with a frozen teacher. We constrain this change by aligning the current final prediction with the same frozen teacher:

$$\begin{aligned} \mathcal{L}_{\mathrm{A}} &= \frac{1}{N} \sum_{i=1}^{N} \mathrm{KL}\big(p_D^\theta(\cdot \mid c_i) \,\big\|\, p_D^\star(\cdot \mid c_i)\big)\,, \\ \mathcal{L}_{\mathrm{LM}} &= \mathcal{L}_{\mathrm{T}} + A\mathcal{L}_{\mathrm{A}}. \end{aligned} \tag{3}$$

Here $A > 0$ controls the strength of the anchor. Anchor constrains predictions on practice contexts;

3.4. Gate: learning when to stop

As practice changes the predictions at different depths, the exit policy must adapt to determine which predictions to use. We train the gate to balance agreement with the current final prediction against the cost of additional computation. For token context c_i, the prediction cost at depth d is

$$\kappa_{i,d} = \mathrm{KL}\big(p_d^\theta(\cdot \mid c_i) \,\big\|\, p_D^\theta(\cdot \mid c_i)\big)\,. \tag{4}$$

Table 1. **Crafter performance and selected depth.** Seen and Unseen denote training and new worlds. Results average seven seeded repetitions; ± denotes their standard deviation. Loops averages per-game mean selected token depth over Seen and Unseen games; readout is dense except in the ragged rows. Bold marks the largest displayed performance mean or smallest depth in each column.

		Ouro-1.4B					Ouro-2.6B				
		Seen		Unseen			Seen		Unseen		
Method	Exit	Ach.↑	Score↑	Ach.↑	Score↑	Loops↓	Ach.↑	Score↑	Ach.↑	Score↑	Loops↓
Base	Gate	1.00±0.00	0.4	1.64±0.05	0.7	3.29	2.04±0.33	1.4	1.89±0.32	1.3	3.30
SFT	Gate	10.81±0.38	22.7	7.19±0.58	12.6	3.63	12.61±0.16	26.0	9.97±0.85	25.3	3.40
GRPO [38]	Gate	5.01±1.82	9.5	4.16±1.16	6.7	3.57	6.45±1.32	14.2	5.50±1.88	10.6	3.36
OPD [1]	Gate	8.83±0.62	15.7	7.17±0.68	11.9	2.93	**12.76**±0.53	29.7	10.67±0.60	24.2	2.90
OPD^2 [17]	Gate	8.11±0.36	14.7	7.01±0.70	10.7	3.17	11.93±0.40	26.7	9.70±1.32	22.3	3.15
OPSA [11]	Gate	2.82±0.87	5.5	2.30±0.85	2.8	3.62	3.44±0.61	7.7	2.93±0.70	4.4	3.42
OPSD [53]	Gate	8.69±0.74	17.2	7.30±0.61	10.2	3.59	8.99±0.61	20.5	7.14±0.97	13.8	3.40
U-OPSD [23]	Gate	7.83±0.63	13.8	6.74±0.58	10.5	3.62	12.11±0.99	29.1	9.60±1.13	20.7	3.39
LoopOPD [45]	Fixed-4	9.08±0.32	18.8	6.67±0.91	11.3	4.00	12.49±0.33	29.0	9.51±0.74	23.4	4.00
FLUENT	Gate	**11.54**±0.50	**26.1**	**8.40**±0.94	**16.5**	2.66	12.69±0.40	**30.8**	**11.31**±1.11	**27.0**	2.58
FLUENT (ragged)	Gate	9.25±1.00	25.5	8.02±0.28	15.1	**2.52**	12.29±0.41	29.5	10.30±0.47	26.7	**2.50**

The gate minimizes

$$\mathcal{L}_{\mathrm{G}} = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \Big[\sum_{d=1}^{D} \pi_\phi(d \mid c_i)(\kappa_{i,d} + \lambda d) - \beta H(\pi_\phi(\cdot \mid c_i)) \Big]. \tag{5}$$

Here $\mathcal{M}$ is the set of response-token positions included in gate training, λ controls the depth penalty, and β weights the entropy H of the exit distribution. The prediction cost favors agreement with the current final loop, while the depth penalty favors earlier exits when agreement is comparable. The entropy term encourages a less concentrated exit distribution during training. The gate reference is the current final prediction p_D^θ, whereas Translate and Anchor use the frozen teacher $p_D^\star$. Anchor connects these two references by limiting how far the current final prediction moves from the teacher. At inference, the learned gate selects the output depth using Eq. (1); costs $\kappa_{i,d}$ are training signals.

Overall training. Combining the model objective in Eq. (3) with the gate objective in Eq. (5), we write

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{T}} + A\mathcal{L}_{\mathrm{A}} + \mathcal{L}_{\mathrm{G}}\big(\phi; \mathrm{sg}[\theta]\big), \tag{6}$$

where sg denotes stop-gradient. The hidden states and prediction costs used for gate training are detached from the model. Thus, θ is updated using $\mathcal{L}_{\mathrm{T}} + A\mathcal{L}_{\mathrm{A}}$, whereas ϕ is updated using $\mathcal{L}_{\mathrm{G}}$. Gate gradients do not modify the shared model. Figure 3 summarizes the full pipeline: the frozen deep teacher supervises shallow predictions and anchors the current final prediction, while the gate learns when to use the updated predictions.

4. Experiments

4.1. Setup

Models and what is trained. In Crafter [14], Ouro-1.4B and Ouro-2.6B [55] use up to four loops. We start from demonstration-trained SFT and update the shared model and gate. In LIBERO [26], we start from public RD-VLA [42], freeze the vision–language and proprioception modules, and train the recurrent action head and a new exit gate. On the real robot, initialization uses five hours of teleoperation and 200 hours of synthetic Isaac Sim data. The primary experiment reuses their input states, with the frozen iteration-eight output as target, without additional robot rollouts. The language experiment trains Ouro-1.4B for 100 steps on the math, science, and code prompt mixture used by OPD^2 [17].

Continuous actions. Our framework also applies to models that produce continuous actions rather than token distributions. We replace KL divergence with MSE in the distillation, anchoring, and gate objectives. Translate transfers deep teacher predictions to shallow iterations, Anchor constrains the prediction at the teacher depth, and Gate balances prediction discrepancy against iteration cost.

RD-VLA setting. For RD-VLA [42], the action head runs for up to $D = 32$ iterations, and the action can be read out after any of them. The teacher is the output of the frozen initial model after eight iterations. Translate distills it into the student's output at iteration two, and Anchor holds the

Table 2. **LIBERO success and executed action-head iterations.** Left: each method's default stopping rule (consecutive-action MSE for baselines, learned gate for FLUENT). Right: four fixed iterations. Evaluation uses 30 new initial states per task and seven repetitions (8,400 episodes per method per exit). ± is the standard deviation of the seven overall rates. Bold/underline mark first/second displayed values. †Uniform supervision includes the final iteration but no extra Anchor penalty.

	Default stopping rule						Success at four iterations (%)				
Method	Spatial	Object	Goal	Long	Avg.	Iters↓	Spatial	Object	Goal	Long	Avg.
RD-VLA [42]	<u>96.7</u>	<u>96.8</u>	93.6	<u>80.4</u>	91.9±0.58	6.27	88.1	92.4	85.8	62.3	82.2±0.71
GRPO [38]	96.9	96.4	93.6	80.4	91.8±0.44	6.25	89.2	93.1	87.3	65.0	83.7±0.66
OPD [1]	**97.1**	96.7	**95.4**	79.4	<u>92.2</u>±0.51	4.78	<u>96.6</u>	<u>97.1</u>	**95.4**	<u>77.8</u>	<u>91.7</u>±0.60
OPD^2 [17]	83.3	75.1	81.9	43.0	70.8±1.92	5.49	77.3	76.4	84.0	38.6	69.1±2.10
OPSA [11]	0.0	0.0	0.0	0.0	0.0±0.00	**3.44**	0.0	0.0	0.0	0.0	0.0±0.00
OPSD [53]	77.7	95.2	9.2	45.1	56.8±2.41	6.56	66.7	94.2	5.4	22.9	47.3±2.77
U-OPSD [23]	69.9	92.1	21.2	64.6	61.9±2.18	6.46	51.6	82.7	14.0	43.3	47.9±2.63
LoopOPD [45]	93.4	95.2	91.3	78.6	89.6±0.72	6.26	85.1	90.1	88.9	70.0	83.5±0.95
FLUENT	96.1	**99.0**	<u>95.0</u>	**83.9**	**93.5**±0.62	<u>3.97</u>	**97.3**	**97.2**	<u>94.5</u>	**80.2**	**92.3**±0.41

Table 3. **Real-robot pick-and-place.** Success is over all attempted episodes; Time and Chunks average successful episodes. Iters is mean executed head iterations per chunk. ± denotes standard deviation across seven repetitions. Episodes use a 300 s timeout.

Method	Succ. (%)↑	Time (s)↓	Chunks	s/chunk	Iters↓
RD-VLA (SFT)	62.29±1.60	69	86	0.80	8.25
OPD	70.43±1.90	42	61	0.69	5.50
w/o Anchor	66.14±2.27	36	58	0.62	3.90
SFT, 32 iterations	65.71±1.98	125	70	1.78	32
SFT, 4 iterations	47.86±2.41	60	97	0.62	4
FLUENT, delayed	69.86±1.86	49	61	0.80	**3.46**
FLUENT	**74.57**±1.72	**31**	52	0.60	**3.46**

student's output at iteration eight to the same target:

$$\mathcal{L}_{\text{act}} = \underbrace{\text{MSE}(a_2^\theta, a_8^\star)}_{\text{Translate}} + \underbrace{3\,\text{MSE}(a_8^\theta, a_8^\star)}_{\text{Anchor}}. \tag{7}$$

Here $a_8^\star$ is the action output of the frozen initial model after eight iterations. The gate uses normalized MSE to the current final action output as its prediction cost and $\gamma d/32$ as its iteration cost in Eq. (5). The suite-specific normalization is calibrated once before training and then fixed.

Baselines. We compare adapted OPD, OPD^2, OPSA, OPSD, U-OPSD, and LoopOPD [1, 11, 17, 23, 45, 53]. For a controlled comparison, we adapt these methods to use deep-loop predictions as teacher targets and shallow-loop predictions as student outputs, without introducing an external teacher model. Within each setting, all methods start from the same initial checkpoint as FLUENT.

4.2. Crafter: higher performance at lower depth

Benchmark and protocol. Crafter is a procedurally generated survival game with 22 distinct achievements [14]. We run ten practice rounds, each containing two games in fixed training worlds and one game in a newly generated world. Only states from the training worlds enter practice. This setting evaluates both improvement in repeatedly encountered worlds and transfer to new ones. Table 1 aggregates play before the update in each round. Seen and Unseen denote training and new worlds, respectively. The main comparison uses dense readout.

Results. FLUENT improves Seen and Unseen achievements over SFT at both model sizes while selecting shallower predictions. On Ouro-1.4B, Unseen achievements increase from 7.19 to 8.40 (16.8%), while mean selected depth decreases from 3.63 to 2.66 (26.7%). Ouro-2.6B shows the same pattern: Unseen achievements increase from 9.97 to 11.31 (13.4%), and mean selected depth decreases from 3.40 to 2.58 (24.1%).

The achievement gains are especially pronounced in new worlds. At both model sizes, the improvement over SFT is larger in Unseen worlds than in Seen worlds. Among the methods evaluated under dense readout, FLUENT achieves the highest Unseen mean achievements and Score, together with the lowest mean selected depth. Methods such as OPD remain competitive in the practice worlds, but FLUENT leads when new-world performance and selected depth are considered together.

These results capture the fluency we seek from practice: the benefits extend beyond the worlds already visited. The practiced policy uses earlier predictions while performing better in new worlds.

Using an earlier prediction does not by itself skip computation in Ouro. Later tokens attend to the key–value cache of earlier tokens at every loop, so under dense readout a token that exits early is still computed to the final loop. To test whether the practiced policy also works when computation is actually skipped, we run inference with Ragged KV [30], under which each generated token stops at its

Table 4. **Ouro-1.4B language evaluation.** Avg is the unweighted mean of twelve accuracy metrics. Loops is the equal-weight mean of generated-token selected depths on MMLU-Pro, BBH, GSM8K, and MATH500. Evaluation uses dense readout. ± denotes standard deviation of the aggregate over seven repetitions; no standard deviation was reported for the fixed pretrained Vanilla reference.

		General						Math		HumanEval		MBPP		Summary	
Method	Exit	MMLU	MMLU-Pro	BBH	ARC-C	HellaSwag	WinoGrande	GSM8K	MATH500	Base	Plus	Base	Plus	Avg	Loops↓
Vanilla	Gate	**68.3**	41.1	**64.9**	59.1	**72.4**	69.1	**69.9**	72.4	70.1	63.4	72.2	**60.8**	65.3	3.05
OPD [1]	Gate	67.9	**41.2**	64.3	58.8	72.2	69.1	68.2	73.4	**71.3**	**65.2**	72.5	60.1	65.3±0.2	3.09
OPD^2 [17]	Gate	66.8	40.4	61.7	57.6	71.5	68.2	63.2	74.0	65.2	59.8	69.8	59.0	63.1±0.5	2.74
OPSA [11]	Fixed-4	67.4	38.9	63.6	**59.4**	71.4	68.0	66.9	70.6	65.9	60.4	69.6	57.1	63.3±0.4	4.00
OPSD [53]	Gate	68.0	40.5	64.4	58.7	72.1	69.0	69.1	73.0	69.5	63.4	72.0	59.9	65.0±0.3	3.07
U-OPSD [23]	Gate	68.2	40.2	64.8	58.9	72.3	69.5	69.8	74.0	70.1	64.6	**73.5**	60.1	**65.5**±0.2	3.03
LoopOPD [45]	Fixed-4	67.1	40.6	63.7	59.2	71.9	68.7	68.8	72.8	67.1	62.8	70.4	58.2	64.3±0.3	4.00
FLUENT	Gate	67.9	40.7	63.5	59.3	72.2	**69.9**	68.8	**74.4**	70.7	**65.2**	71.7	60.6	65.4±0.2	**2.48**

selected loop. This additional execution setting reaches the lowest mean selected depth in Tab. 1, with new-world achievements above SFT at both model sizes (8.02 vs. 7.19; 10.30 vs. 9.97).

4.3. Real robot: faster and more reliable

Task. A 6-DoF arm moves specified objects to designated locations. The 100 scenarios vary the number (one to seven), shapes, and positions of objects, with one to three objects specified for relocation. Evaluation layouts are excluded from training. An episode succeeds when all specified objects reach their target locations. We repeat each scenario seven times. Completion time is measured on the robot from episode start to success and averaged over successful episodes. An episode that does not succeed within 300 s is counted as a failure.

Results. FLUENT improves both task success and completion time. Success increases from 62.29% for SFT to 74.57%, while mean completion time over successful episodes decreases from 69 s to 31 s, a 2.23× ratio. Executed action-head iterations decrease by 58.1%, from 8.25 to 3.46. Successful episodes also use fewer action chunks on average, from 86 to 52, while mean time per chunk decreases from 0.80 s to 0.60 s.

The success gain persists when action timing is matched. Delaying FLUENT to the SFT cadence of 0.80 s per chunk still yields 69.86% success, compared with 62.29% for SFT. Without the added delay, FLUENT reaches 74.57%. Meanwhile, SFT with four fixed iterations has a similar per-chunk execution time but lower success.

4.4. LIBERO: success with fewer iterations

Benchmark and protocol. LIBERO comprises Spatial, Object, Goal, and Long suites [26]. We distill teacher predictions at iteration eight into student predictions at iteration two. We train suite-specific action heads for three rounds on recorded policy inputs. At evaluation, FLUENT uses its learned gate to select the number of executed iterations. Each method is evaluated on 30 new initial states for each of 40 tasks over seven repetitions, giving 8,400 episodes per stopping rule. Table 2 reports results under the default stopping rule of each method and with four fixed iterations.

Results. The initial RD-VLA policy already achieves approximately 92% success. Our primary focus is therefore how few iterations can sustain high control performance, with additional success gains reported as a secondary benefit. FLUENT achieves 93.5±0.62% success at 3.97 iterations per chunk, compared with 91.9±0.58% at 6.27 for RD-VLA. It thus executes 36.7% fewer action-head iterations while achieving a higher mean success rate.

The fixed-four evaluation makes the improvement at shallower depth particularly clear. The original RD-VLA head achieves 82.2% success, whereas FLUENT reaches 92.3% at the same iteration count: a 10.1-percentage-point improvement. FLUENT also achieves the highest fixed-four mean among the compared methods, including OPD at 91.7%. Practice therefore does more than advance the stopping point: it enables earlier iterations to produce actions that support high task success.

4.5. Language models: earlier outputs after fixed-corpus distillation

Benchmark and protocol. This experiment tests whether useful predictions can become available at shallower depth using a fixed prompt corpus, without collecting new environment interactions. We train Ouro-1.4B for 100 steps on an equal mixture of math, science, and code prompts. The deep prediction of the initial model provides the teacher, and distillation uses response contexts generated by the shallow student.

We evaluate general knowledge and reasoning, mathematics, and code generation using MMLU, MMLU-Pro, BBH, ARC-C, HellaSwag, WinoGrande, GSM8K, MATH500, HumanEval/+, and MBPP/+ [2, 5, 8, 9, 15, 16, 25, 27, 36, 41, 44, 51]. Mean accuracy gives equal weight to the twelve evaluation metrics. Mean selected depth

averages generated-token depth across MMLU-Pro, BBH, GSM8K, and MATH500. Evaluation uses dense readout.

Results. FLUENT maintains the mean accuracy of the initial vanilla model while using shallower predictions (Table 4). Mean accuracy is 65.4%, compared with 65.3% for the vanilla model, while mean selected depth decreases from 3.05 to 2.48, a reduction of 18.7%. Among the compared methods, FLUENT achieves the lowest mean selected depth while retaining the original level of mean accuracy.

The baseline comparison highlights the importance of considering accuracy and selected depth together. OPD reaches 65.3% mean accuracy, but its selected depth remains 3.09. U-OPSD reaches 65.5% at a selected depth of 3.03. By comparison, FLUENT uses predictions at a substantially shallower mean depth of 2.48 with comparable mean accuracy. OPD^2 also reduces selected depth, to 2.74, but its mean accuracy falls to 63.1%.

These results extend FLUENT beyond interactive environments. Even with a fixed set of prompts, cross-depth distillation and exit-policy updates enable the model to use earlier predictions while preserving mean accuracy.

4.6. Ablation study

Component ablation. Translate, Anchor, and Gate play complementary roles (Table 5). Translate alone reduces mean selected depth from 3.63 for SFT to 2.97, while unseen achievements change from 7.19 to 7.00. Adding Anchor raises mean achievements to 8.36.

Gate retraining makes the improved performance available at shallower depth. Adding Gate to Translate and Anchor reduces mean selected depth from 3.24 to 2.66, with 8.40 mean achievements. At the same rounded selected depth of 2.66, the no-Anchor variant earns 7.05 achievements. This comparison highlights the importance of combining early exit with prediction preservation.

Gate-only retraining also helps, reaching 7.70 achievements at 3.15 loops. The full method combines the highest achievement mean with the joint-lowest selected depth. Together, shallow-prediction learning, deep-prediction preservation, and exit-policy updates enable FLUENT to achieve higher task performance at shallower depth.

Change over practice. As practice progresses, FLUENT uses shallower predictions while improving task performance (Fig. 4). Selected depth decreases sharply during the first rounds and more gradually thereafter, following the qualitative pattern illustrated in Fig. 1. Meanwhile, cumulative Score improves in both training and new worlds. Practice therefore makes the policy more proficient with fewer loops, including in environments it has not practiced in. Refreshing experience also helps: at equal update counts, collecting new states each round yields a mean selected depth of 2.66, compared with 2.89 when repeatedly training on first-round states.

Table 5. **Effects of Translate, Anchor, and Gate in Crafter** (Ouro-1.4B). Checkmarks indicate enabled training components; Gate denotes exit-policy retraining. Achievements and Score concern Unseen worlds; Loops averages selected token depth over Seen and Unseen games under dense readout. Bold marks the best displayed means (higher Ach. and Score, lower Loops).

Training			Unseen		
Translate	Anchor	Gate	Ach.	Score	Loops
–	–	–	7.19±0.58	12.6	3.63
–	–	✓	7.70±0.46	14.7	3.15
✓	–	–	7.00±0.34	11.5	2.97
✓	–	✓	7.05±0.86	11.7	**2.66**
✓	✓	–	8.36±0.73	**16.5**	3.24
✓	✓	✓	**8.40**±0.94	**16.5**	**2.66**

Table 6. **Per-loop Crafter analysis** (Ouro-1.4B, unseen worlds). All rows evaluate final checkpoints with matched world sets; parentheses give mean gate-selected depth. Bold marks the highest achievement mean in each row.

Exit	SFT	w/o Anchor	FLUENT
Loop 1	0.00±0.00	**4.86**±0.55	4.62±0.47
Loop 2	1.02±0.21	**6.74**±0.71	5.08±0.44
Loop 3	6.10±0.52	**7.86**±0.80	7.71±0.58
Loop 4	7.12±0.60	6.31±0.77	**8.07**±0.86
Gate	7.21±0.55	7.05±0.86	**8.24**±0.73

Per-loop analysis. Improving shallow predictions does not automatically improve every later loop or the performance obtained through adaptive exits (Table 6). Without Anchor, distillation improves the first three loops over SFT, but fourth-loop performance falls from 7.12 to 6.31 achievements. Moreover, gated play achieves 7.05, below the 7.86 achievements available at the third loop. The improvement in intermediate predictions is therefore not fully reflected in gated performance.

In contrast, FLUENT improves all four fixed-depth readouts over SFT and achieves 8.24 achievements with its learned exit policy, compared with 7.21 for SFT. Its gated performance also exceeds its strongest fixed-depth readout, which achieves 8.07. These results illustrate why practice must address both the predictions and their selection: better shallow predictions alone are insufficient; the policy must also learn when to use them.

Repeated practice. We test whether the gains of practice hold under repeated updates: ten cycles of 10 unlabeled practice episodes followed by one update, on scenarios disjoint from the evaluation scenarios (Tab. 7). FLUENT keeps its success rate (74.57% to 74.21%) while its executed iterations fall further (3.46 to 3.21). OPD loses 5.48 points (70.43% to 64.95%) with unchanged iterations. A

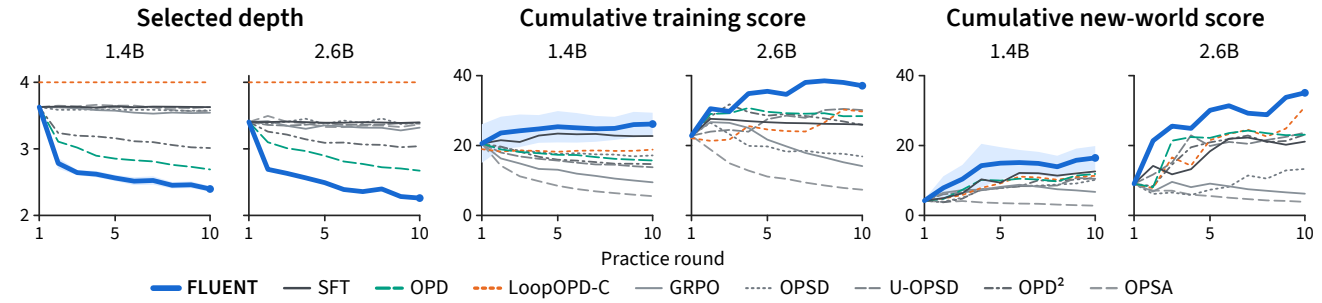

Figure 4. **Depth and cumulative Crafter score over practice.** Curves show seven-run means at 1.4B (FLUENT shading: ±1 SD) and a single-run illustration at 2.6B. Seven-repetition aggregate results at both sizes appear in Table 1. Selected depth and cumulative score have different aggregation rules.

Table 7. **Repeated practice on the real robot.** The policy alternates between 10 practice episodes and one update, with no labels and no reward. Starting policies and fixed evaluation scenarios are those of Tab. 3.

	Starting policy		Repeated practice	
Method	Succ. (%)	Iters	Succ. (%)	Iters
OPD	70.43±1.90	5.50	64.95±1.42	5.50
FLUENT	74.57±1.72	3.46	74.21±0.97	3.21

single round of deep-to-shallow distillation is useful on its own: OPD raises success over SFT from 62.29% to 70.43% (Tab. 3), but this benefit does not persist under repeated updates. Anchor regularizes FLUENT toward the fixed initial model, and on a fixed context set the bound on this drift does not grow with the number of rounds, provided the objective does not exceed its initial value.

5. Discussion

The ruler moves with what it measures. Early-exit methods usually treat the last layer as a fixed target. In a looped model this fails: all loops share weights, so training the first loop also changes the final loop that serves as its ruler. Without Anchor, the model has a better prediction at its third loop, yet its gated play stays below it (7.05 vs. 7.86; Tab. 6). Having a good prediction and choosing to use it are separate problems. Protecting the reference is not free: the more strongly it is protected, the less the first loop can adapt.

The student is not a copy. A model that learns from itself, with no labels or rewards, ends up better than its teacher (fourth loop 8.07 vs. 7.12; Tab. 6). A weight-shared model cannot memorize every situation separately, so it learns the teacher's judgment pooled over similar situations, which filters out the teacher's occasional mistakes, while Anchor largely preserves its confident decisions. Practice can correct the mistakes a teacher makes occasionally, but it cannot teach what the teacher does not know.

Fluent is not the same as better. Practice lowered depth in every setting, but it did not always raise performance: Crafter and the real robot gained both, while LIBERO, which starts from about 92% success, and the language models gained mainly depth. We hypothesize that two conditions matter: whether the initial model leaves room to improve, and whether the practiced situations resemble those at evaluation. Designing practice is then less a question of how much to practice than of what to practice.

Limitations and future directions. The initial model provides a fixed teacher, so the quality of its predictions remains a constraint on the supervision available during practice. One way forward is to split practice into phases and refresh the teacher between them, while keeping it fixed within a phase so that the reference of the gate stays stable. Another is to give each loop a small number of its own parameters, to explore whether this eases the trade-off between adapting the first loop and preserving the final loop. Finally, extending the approach beyond looped models is a promising direction. Multi-exit transformers also share layers across exits [37, 47], so updates to shared layers can affect both early and final predictions, motivating joint study of prediction preservation and exit learning.

6. Conclusion

The same task is hard the first time and easy once it is familiar. We started from the view that difficulty depends not only on the input but also on experience. FLUENT is a self-distillation procedure that teaches looped models to produce useful predictions earlier and to decide when to use them. Translate uses the initial model's final-loop prediction as a frozen teacher for shallow predictions, Anchor regularizes the final-loop prediction toward the same teacher to stabilize the reference of Gate, and Gate adapts the exit policy to the updated predictions. No additional task labels or rewards are required. Across Crafter, LIBERO, real-robot manipulation, and language-model evaluation, FLUENT improves the performance–depth trade-off; on the real

robot, this translates into higher success and shorter successful episodes. Practice can improve both predictions and the depth at which a model learns to use them. For models that repeatedly perform the same tasks, learning from experience to reduce computation is a practical requirement for efficient deployment. A robot that picks up the same cup for the hundredth time should not need to compute as long as it did the first time.

LLM Usage Statement

We used OpenAI Codex and Anthropic Claude to translate the completed manuscript from a non-English language into English and to shorten sentences. We also used these models to assist with the mathematical analysis: they helped formulate the statements and draft the proofs. The authors examined each proof strategy, checked the results and references that the proofs rely on, and verified the correctness of every statement and proof. The authors take full responsibility for the final manuscript, including its technical claims, proofs, references, experimental results, and interpretations.

References

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, pages 21246–21263, 2024. 2, 4, 5, 6

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. 6

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. 1

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 1

[5] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 6

[6] Peter Baile Chen, Yi Zhang, Dan Roth, Samuel Madden, Jacob Andreas, and Mike Cafarella. Log-augmented generation: Scaling test-time reasoning with reusable computation. In *International Conference on Learning Representations*, pages 3339–3358, 2026. 1, 2

[7] Saehun Chun, Wonje Choi, Sera Choi, Sanghyun Ahn, and Honguk Woo. Functional cache grafting: Robust and rapid code-policy synthesis for embodied agents. In *Proceedings of the 43rd International Conference on Machine Learning*, pages 20799–20839, 2026. 1, 2

[8] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. 6

[9] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 6

[10] Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. In *International Conference on Learning Representations*, 2025. 2

[11] Yi Ding and Ruqi Zhang. Does on-policy distillation really distill? from noisy teacher to self-improvement. *arXiv preprint arXiv:2608.31046*, 2026. 2, 4, 5, 6

[12] Wei Fan, Qingyue Yang, Chao-Hong Tan, Qian Chen, Wen Wang, Xiangang Li, Kejiang Chen, Weiming Zhang, and Nenghai Yu. LoOPSD: On-policy self-distillation with looped teachers. *arXiv preprint arXiv:2610.18162*, 2026. 3

[13] In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6: 325–338, 2024. 1, 2

[14] Danijar Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021. 2, 4, 5

[15] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. 6

[16] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of*

the Neural Information Processing Systems Track on Datasets and Benchmarks, 2021. 6

[17] Byeongho Heo, Jaehui Hwang, Sangdoo Yun, and Dongyoon Han. On-policy delta distillation. *arXiv preprint arXiv:2607.15161*, 2026. 2, 4, 5, 6

[18] Riccardo Andrea Izzo, Rimvydas Rubavicius, Gianluca Bardaro, Subramanian Ramamoorthy, Matteo Matteucci, and Alessandro Suglia. Decoupled early exits for task-dependent compute allocation in flow-matching VLAs. *arXiv preprint arXiv:2609.29382*, 2026. 2

[19] Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. LoopFormer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. *arXiv preprint arXiv:2602.11451*, 2026. 1, 2

[20] Boseong Jeon, Yunho Choi, and Taehan Kim. Shallow-π: Knowledge distillation for flow-based VLAs. *arXiv preprint arXiv:2601.20262*, 2026. 2

[21] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 1

[22] Dianqiao Lei and Lianlei Shan. Think less, act early: Reinforced latent reasoning with early exit in vision-language-action models. *arXiv preprint arXiv:2606.15099*, 2026. 2

[23] Yijiang Li, Bingyang Wang, Yijun Liang, Yunjie Tian, Di Fu, and Nuno Vasconcelos. On-policy self-distillation without any supervision. *arXiv preprint arXiv:2608.06296*, 2026. 2, 4, 5, 6

[24] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. 2

[25] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. 6

[26] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023. 2, 4, 6

[27] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, 2023. 6

[28] Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. FastBERT: A self-distilling BERT with adaptive inference time. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6035–6044, 2020. 1, 2

[29] Gordon D Logan. Shapes of reaction-time distributions and shapes of learning curves: a test of the instance theory of automaticity. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18 (5):883, 1992. 1

[30] Aayush Mishra, Arnau Padrés Masdemont, Victor Conchello Vendrell, Jordi Ros Giralt, Arash Behboodi, and Fabio Valerio Massoli. Enabling dynamic computation in looped LMs. *arXiv preprint arXiv:2610.09013*, 2026. 2, 5

[31] Allen Newell and Paul S Rosenbloom. Mechanisms of skill acquisition and the law of practice. In *Cognitive Skills and Their Acquisition*, pages 1–55. Psychology Press, 2013. Reprint of the volume first published in 1981. 1

[32] Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, et al. SkillOS: Learning skill curation for self-evolving agents. *arXiv preprint arXiv:2605.06614*, 2026. 1, 2

[33] Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, pages 94327–94354, 2026. 1, 2

[34] Bo Pan and Liang Zhao. Can past experience accelerate LLM reasoning? *arXiv preprint arXiv:2505.20643*, 2025. 2

[35] Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1355–1364, 2019. 2

[36] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8732–8740, 2020. 6

[37] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35: 17456–17472, 2022. 1, 2, 8

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 4, 5

[39] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion:

Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023. 1, 2

[40] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. 2

[41] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023. 6

[42] Yalcin Tur, Jalal Naghiyev, Haoquan Fang, Wei-Chuan Tsai, Jiafei Duan, Dieter Fox, and Ranjay Krishna. Recurrent-Depth VLA: Implicit test-time compute scaling of vision-language-action models via latent iterative reasoning. *arXiv preprint arXiv:2602.07845*, 2026. 1, 2, 4, 5

[43] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 1, 2

[44] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, 2024. 6

[45] Yi Wang, Rui Qian, Yu Li, Haoyang Yao, and Wenjie Wang. Recurrent self-improvement: Dynamic cross-loop on-policy distillation for looped language models. *arXiv preprint arXiv:2610.10623*, 2026. 2, 4, 5, 6

[46] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 63897–63911, 2025. 2

[47] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 2246–2251, 2020. 1, 2, 8

[48] Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 37:113519–113544, 2024. 2

[49] Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the twentieth European conference on computer systems*, pages 94–109, 2025. 1, 2

[50] Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024. 1, 2

[51] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019. 6

[52] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019. 2

[53] Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026. 2, 4, 5, 6

[54] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. SGLang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024. 2

[55] Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025. 1, 2, 4