Modern Recursive Transformer

Taekhyun Park

Department of Data Science,

Pusan National University, Busan

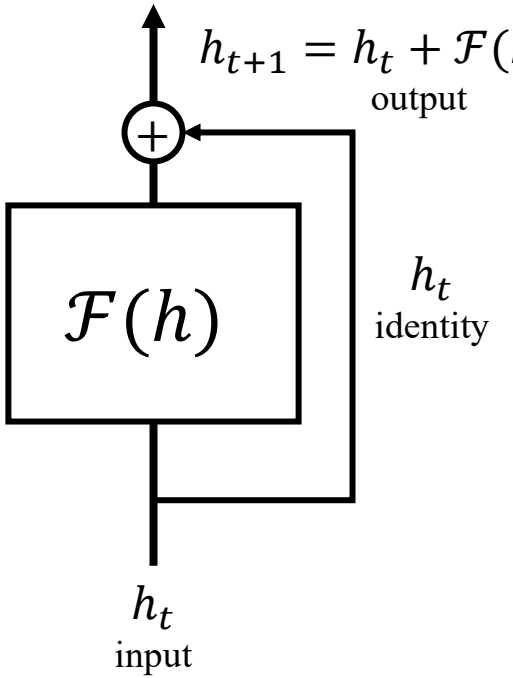
pthpark1@pusan.ac.kr

Contents

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Recursive Deep Learning Model

- Since the advent of **ResNet** (He et al., 2015), deep learning models can be interpreted as **continuous-time dynamical systems**. As training aligns hidden representations within a shared latent space with a basis $\mathcal{B}$, **recursion design** becomes critical to performance and stability(Liu et al., 2026).



Discrete formulation :

$$h_{t+1} = h_t + \mathcal{F}_{\theta_t}(h_t)$$

Continuous formulation :

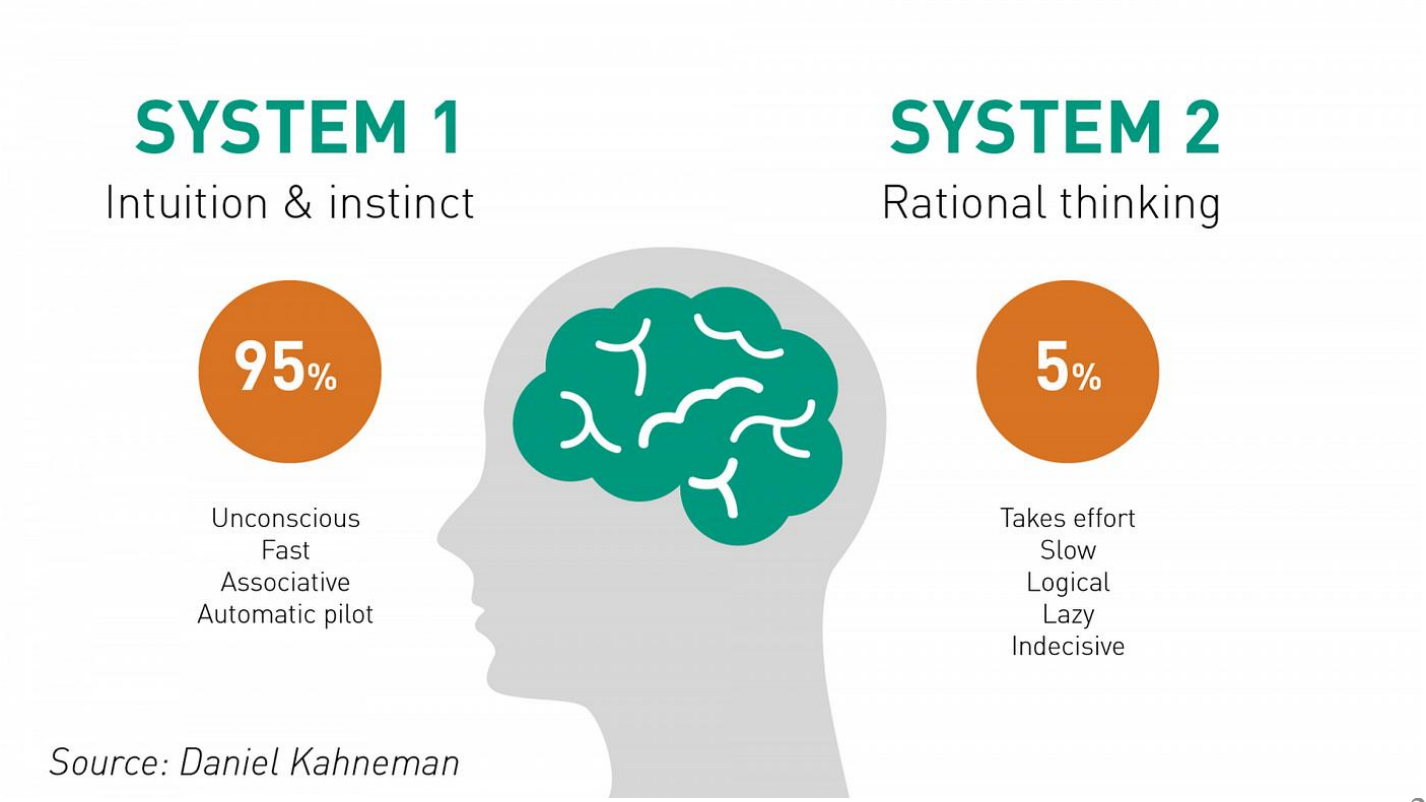
$$\frac{dh(t)}{dt} = F_\theta(h(t), t)$$

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

How to make good Recursive systems for Transformer

1. Scaling of Transformer

- Human cognition is described as two systems: **System 1**, which enables fast(30~100Hz, fast gamma waves), **intuitive thinking** and **System2**, which supports slow(4~8Hz, slow theta waves), **deliberate reasoning** (Kahneman, 2011).

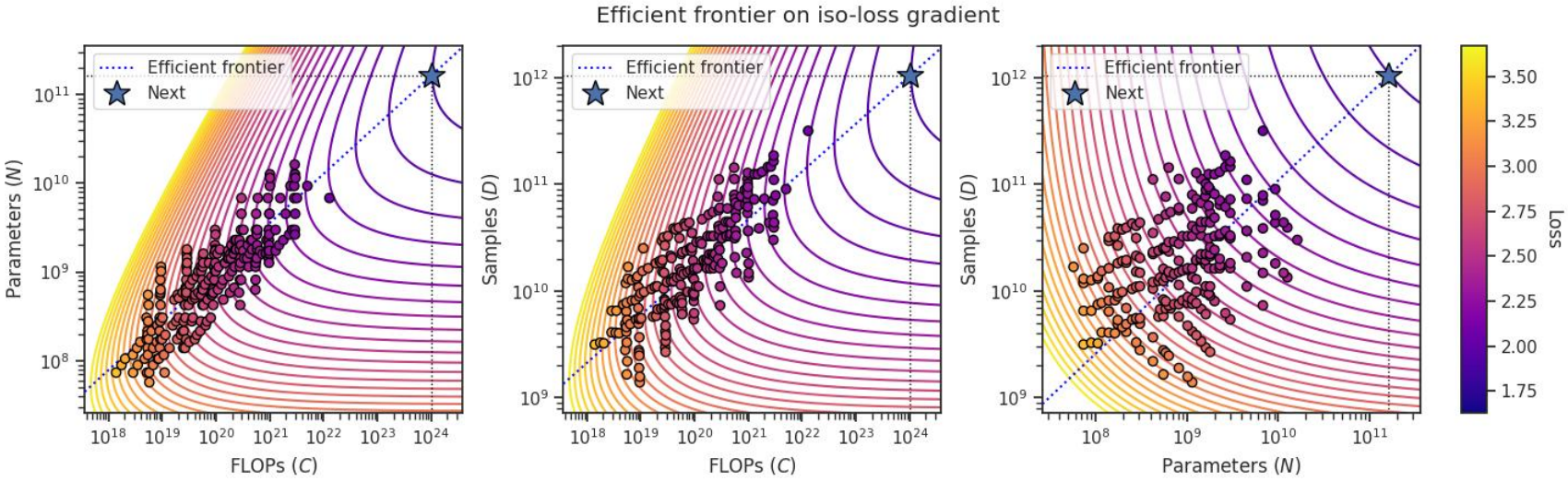


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

How to make good Recursive systems for Transformer

1. Scaling of Transformer

- Inspired by this distinction, Transformer research emphasizes reasoning depth, either by increasing **Test-Time Compute**(TTC), spending more computation per token (Snell et al., 2025) and large-scale training such as **Chinchilla scaling** (Hoffmann et al., 2022).
- Data scarcity limits further scaling, and recent studies show that **TTC** effectively increase recursive reasoning depth, with explicit mechanisms such as **CoT** (Wei et al., 2022) yielding stronger performance gains beyond a certain scale(Snell et al., 2025).

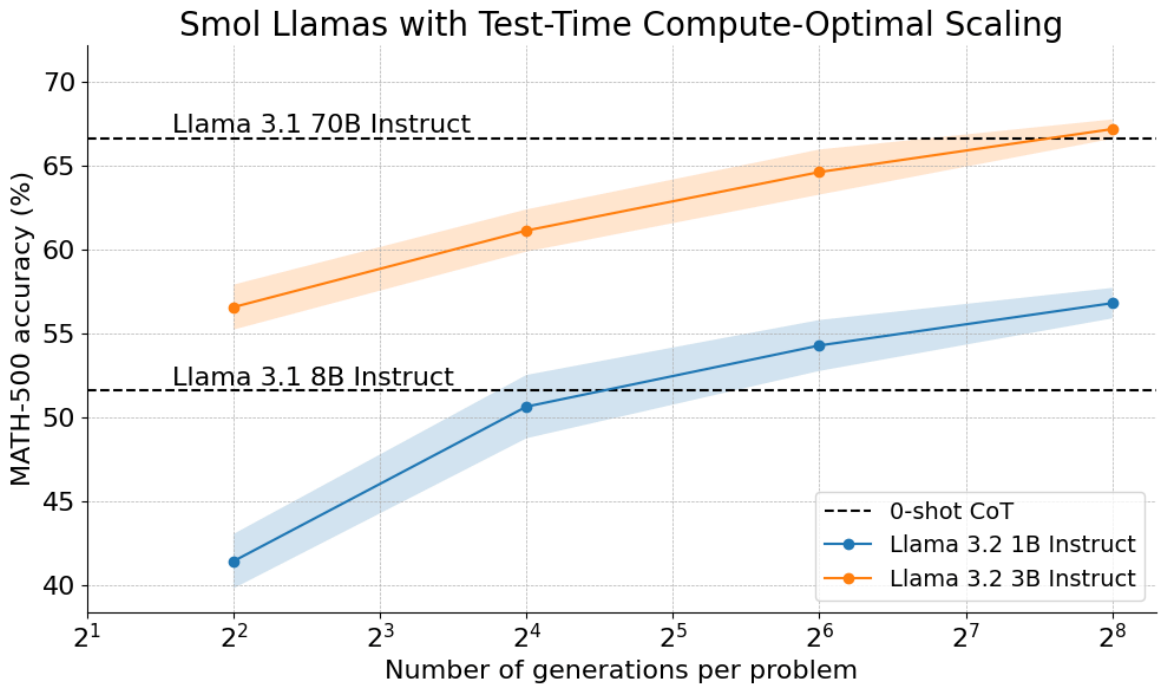


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

How to make good Recursive systems for Transformer

1. Scaling of Transformer

- Inspired by this distinction, Transformer research emphasizes reasoning depth, either by increasing **Test-Time Compute**(TTC), spending more computation per token (Snell et al., 2025) or by large-scale training such as **Chinchilla scaling** (Hoffmann et al., 2022).
- Data scarcity limits further scaling, and recent studies show that **TTC** effectively increase recursive reasoning depth, with explicit mechanisms such as **CoT** (Wei et al., 2022) yielding stronger performance gains beyond a certain scale(Snell et al., 2025).

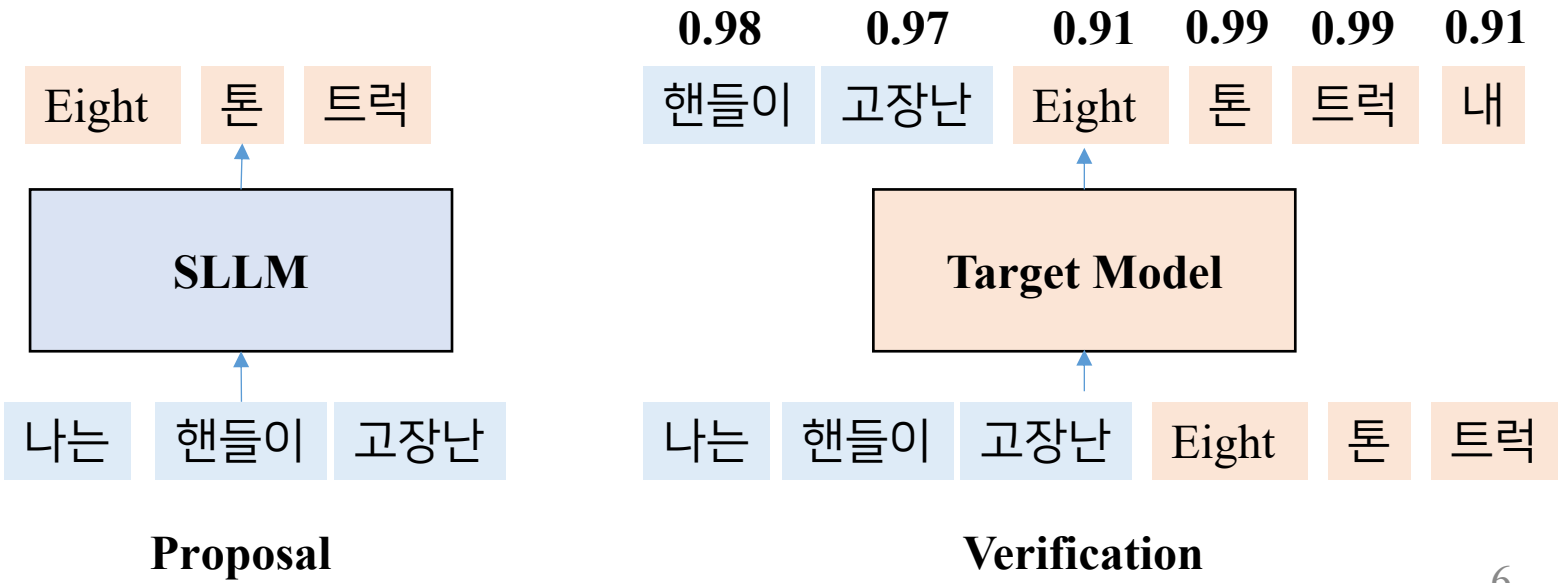


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

How to make good Recursive systems for Transformer

2. Scalable Learning

- However, the performance gains achieved through TTC and Chinchilla-style scaling come with clear trade-offs in computational cost and inference latency.
  - **Speculative Decoding**(Leviathan et al., 2023), first generates candidate outputs with a smaller model and then verifies them with a larger model; if verification fails, the large model recomputes the corresponding parts. (**Google Gemini 3.0**)
  - **Routing-based methods** use lightweight classifiers to route easy queries to smaller models and harder queries to more powerful ones. (**OpenAI GPT5**)
  - **Automated approaches** enable models to dynamically scale their own recursion depth during inference, as seen in architectures such as **Hierarchical Reasoning Model (HRM)**(Wang et al., 2025) and **Energy-based Transformer**(Gladstone et al., 2026).

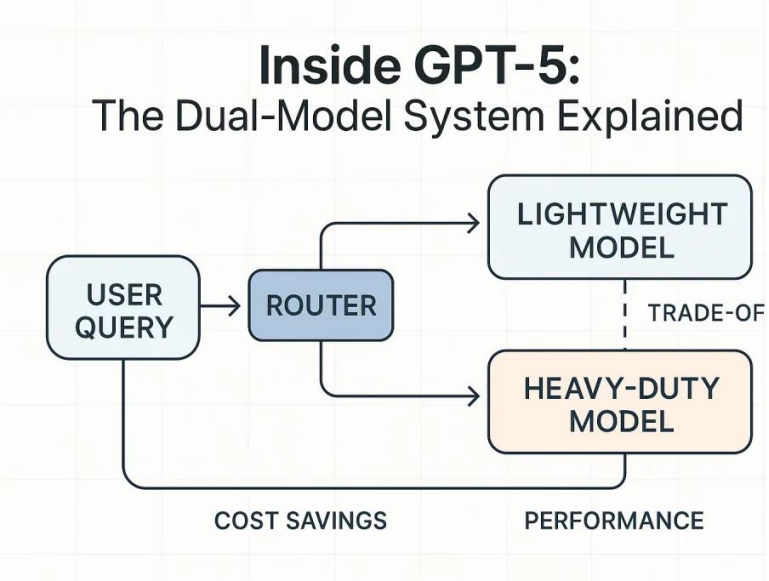


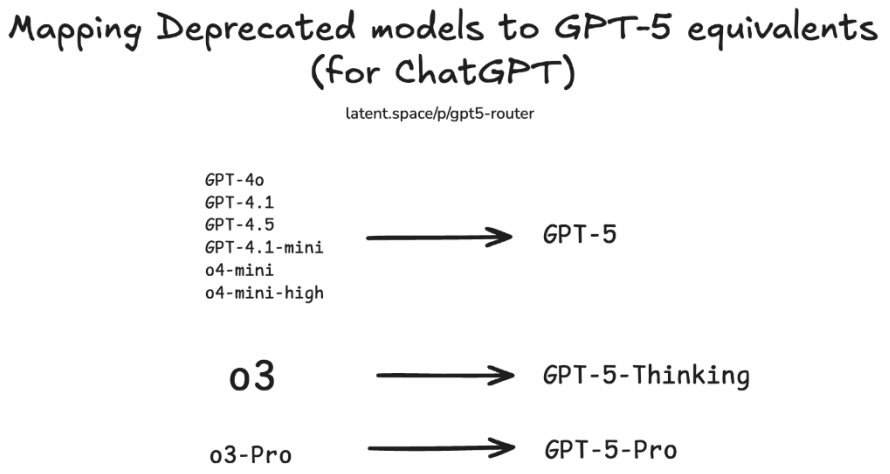
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

How to make good Recursive systems for Transformer

2. Scalable Learning

- However, the performance gains achieved through TTC and Chinchilla-style scaling come with clear trade-offs in computational cost and inference latency.
  - **Speculative Decoding**(Leviathan et al., 2023), first generates candidate outputs with a smaller model and then verifies them with a larger model; if verification fails, the large model recomputes the corresponding parts. (**Google Gemini 3.0**)
  - **Routing-based methods** use lightweight classifiers to route easy queries to smaller models and harder queries to more powerful ones. (**OpenAI GPT5**)
  - **Automated approaches** enable models to dynamically scale their own recursion depth during inference, as seen in architectures such as **Hierarchical Reasoning Model (HRM)**(Wang et al., 2025) and **Energy-based Transformer**(Gladstone et al., 2026).



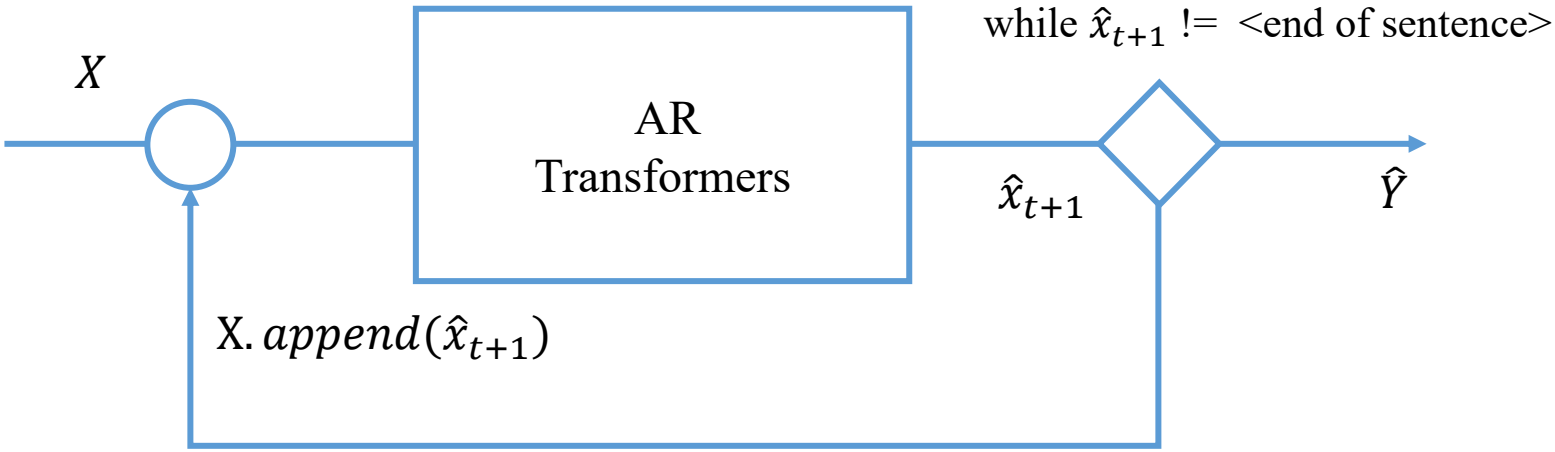


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Autoregressive based Transformer

- To expand **TTC**, GRPO(Shao, Zhihong, et al., 2024) is used to induce CoT reasoning, leading to improved performance.

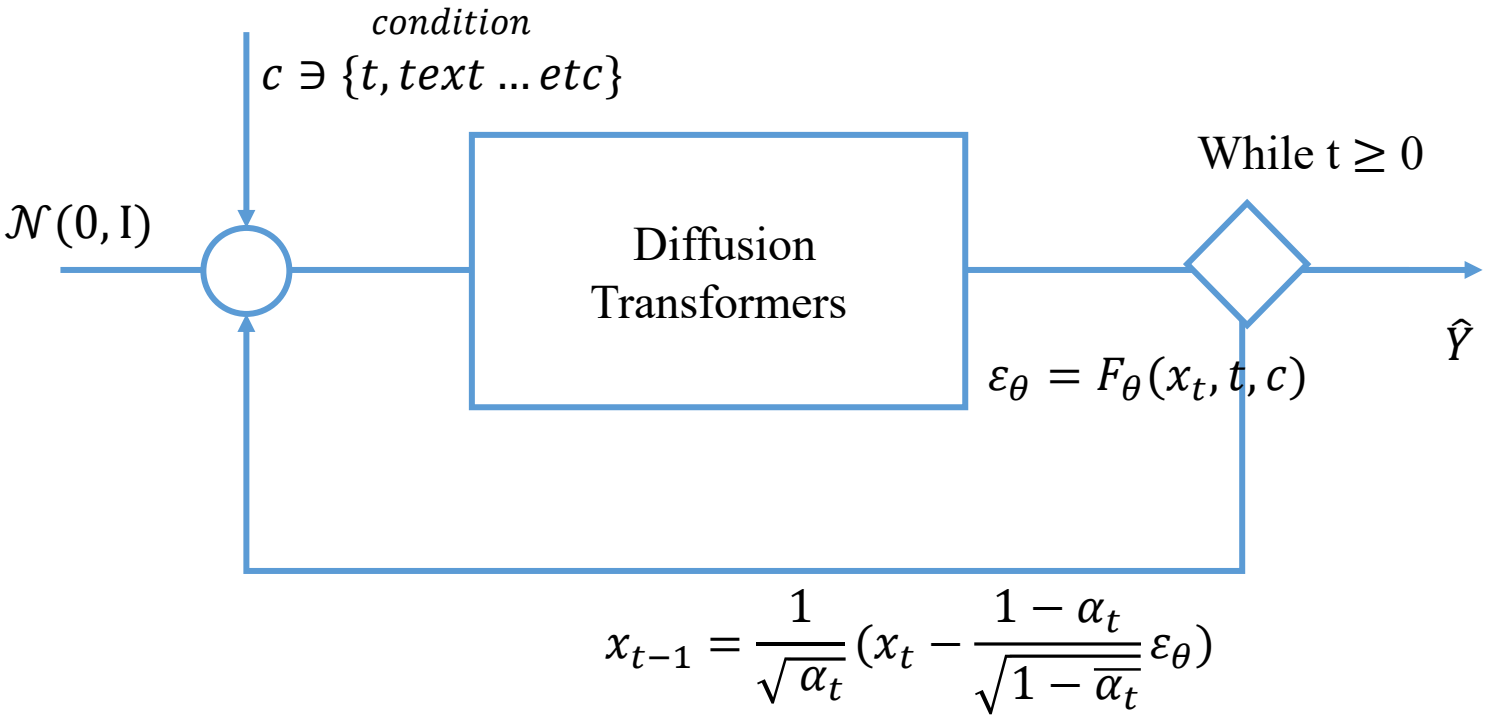
$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}, \quad (3)$$



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Diffusion based Transformer

- Diffusion Transformer(DiT)(Peebles et al., 2023, Nie et al, Neurips 2025) defines generation as a diffusion process based on Brownian-motion-driven SDEs , where the model **estimates the assumed noise component** ε_θ and **iteratively removes** it according to a predefined noise schedule at each step to recover clean representations.

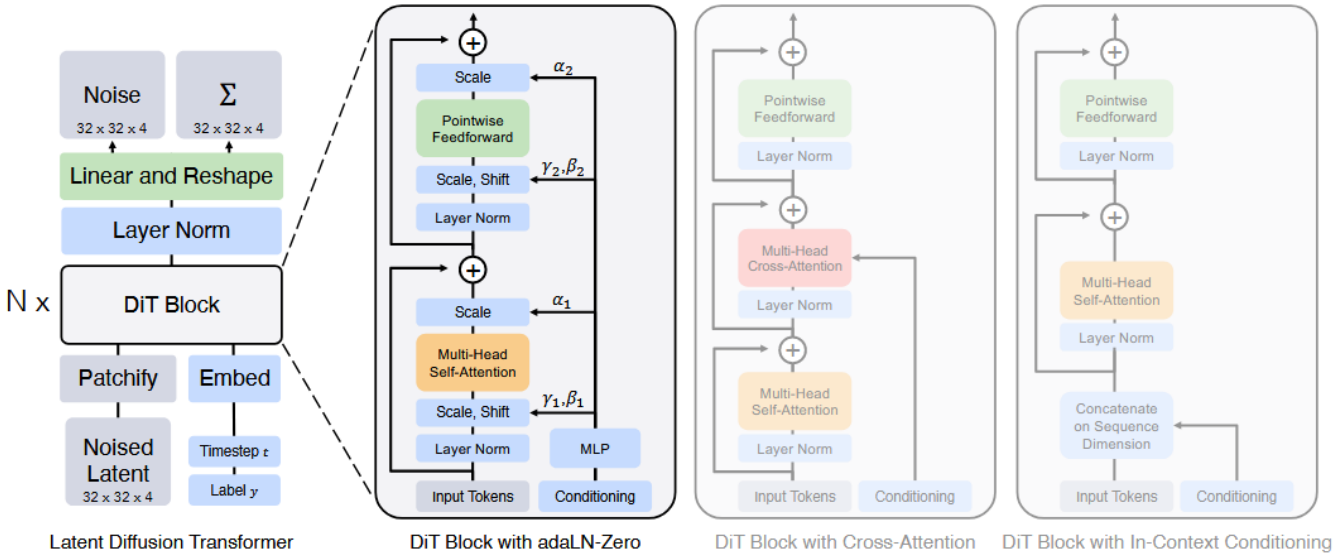


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Diffusion based Transformer

Scalable Diffusion Models with Transformers(ICCV 2023) (Peebles et al., 2023)

- Forward process: $x_t = \sqrt{\overline{\alpha_t}}x_o + \sqrt{1-\alpha_t}\epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$
- Loss (training): $\mathcal{L}(\theta) = -p(x_0|x_1) + \sum_t D_{KL}(q^*(x_{t-1}|x_t, x_0)|p_\theta(x_{t-1}|x_t)) \rightarrow \mathcal{L}_{simple}(\theta) = ||\epsilon_\theta(x_t) - \epsilon_t||_2^2$
- Classifier-free guidance(CFG): $\hat{\epsilon}(x_t, t, c) = \epsilon_\theta(x_t, t, \emptyset) + s(\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, \emptyset))$



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Diffusion based Transformer

Large Language Diffusion Models (Nie et al., 2025, Neurips 2025 Oral)

- Forward process: $q_{t|0}(x_t|x_0) = \prod_{i=1}^{L} q_{t|0}(x_t^i|x_0^i)$
- $q_{t|0}(x_t^i|x_0^i) = \begin{cases} 1-t, x_t^i = x_0^i, \\ t, x_t^i = M, M \text{ denotes the mask token} \end{cases}$
- Loss (training) : $\mathcal{L}(\theta) = -\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L} 1[x_t^i = M] log p_\theta(x_0^i|x_t)\right]$

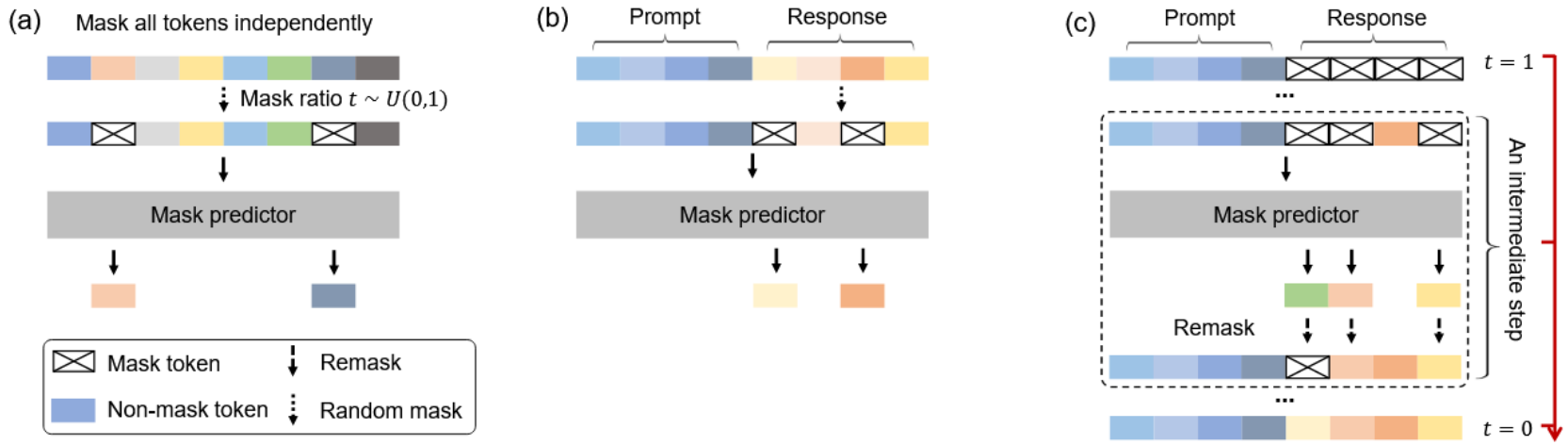
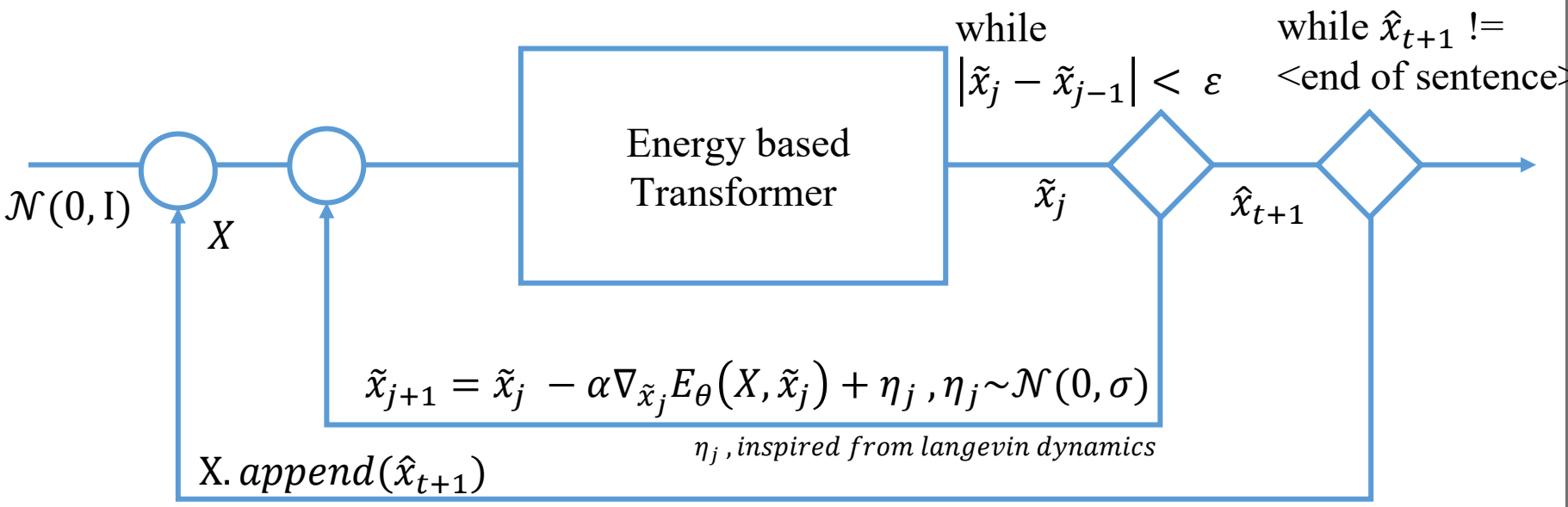


Figure 2. **A Conceptual Overview of LLaDA.** (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked) to $t = 0$ (unmasked), predicting all masks simultaneously at each step with flexible remask strategies.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Energy-Based Transformer

- Energy-Based Transformers(Gladstone, Alex et al, 2026) perform energy minimization in a locally **convex representation space** and **automatically stop upon convergence**, enabling scalable unsupervised System 2 Thinking



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Energy-Based Transformer

- ENERGY-BASED TRANSFORMERS ARE SCALABLE LEARNERS AND THINKERS (Gladstone, Alexi et al. , ICLR 2026 Oral)

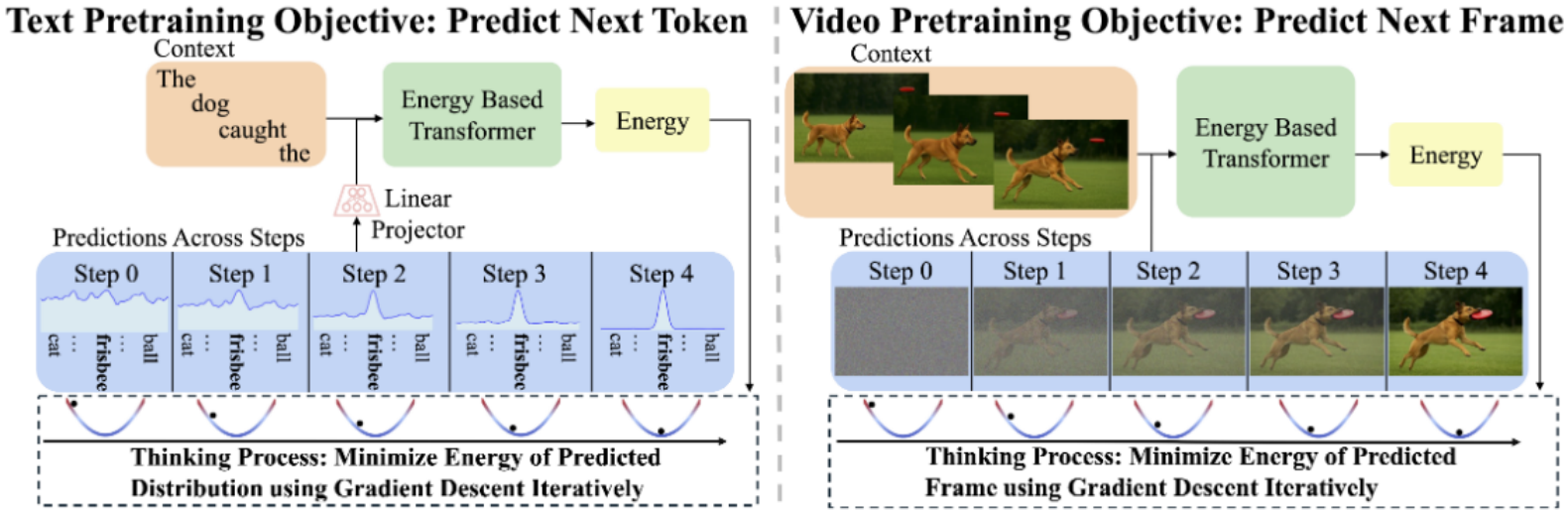
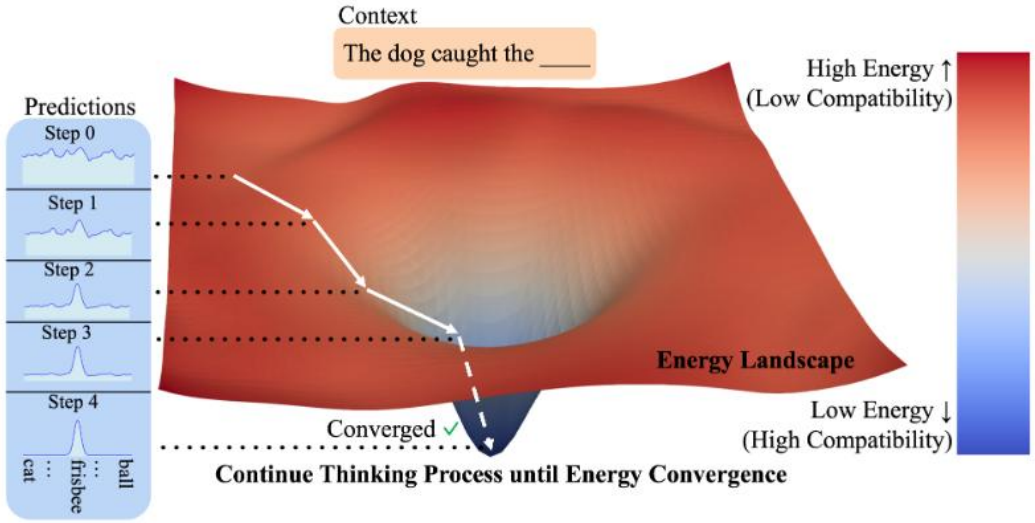


Figure 2: **EBT for Autoregressive Modeling.** Each blue box corresponds to a different prediction at each step of the thinking process, where the initial prediction starts as random. At each step, a new prediction is fed into the model, which gives an energy scalar for the prediction's current *compatibility* (unnormalized likelihood) with the context (Facet 2). Then, the gradient of this energy with respect to the prediction is calculated and used to update the prediction. This gradient descent update is done iteratively to refine the prediction until convergence of the predicted energy, which allows for dynamic use of computation (Facet 1).

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Energy-Based Transformer

- ENERGY-BASED TRANSFORMERS ARE SCALABLE LEARNERS AND THINKERS (Gladstone, Alexi et al. , ICLR 2026 Oral)



Algorithm 1: Training

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

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

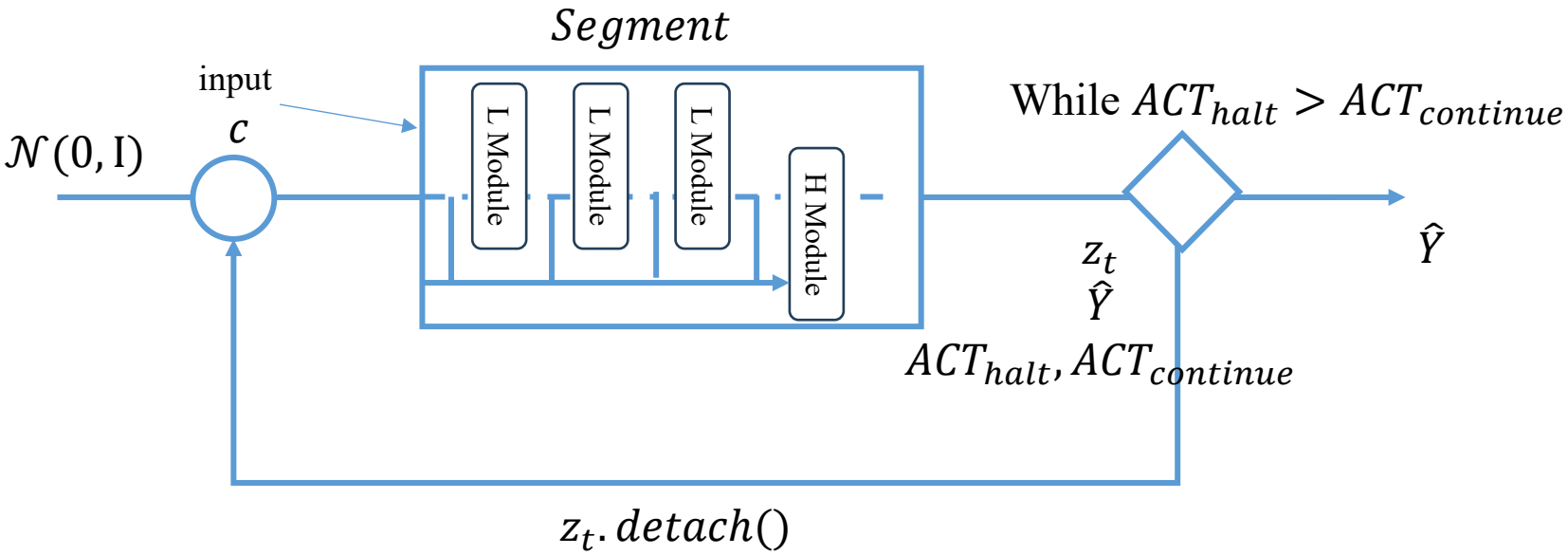
Algorithm 2: Inference with Verification

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

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

Hierarchical Reasoning Model

- Inspired by brain dynamics, HRM was designed two Transformer-based modules : **H module** (slower timescales like theta waves) and **L module** (fast timescales like gamma waves)
- **Deep supervision, 1-step Approximate gradient** and **Adaptive Computational Time(ACT)**, the model learns dynamic thinking trajectories end-to-end, without requiring fixed noise schedules(DiT) or predefined convexity assumptions(EBT).



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

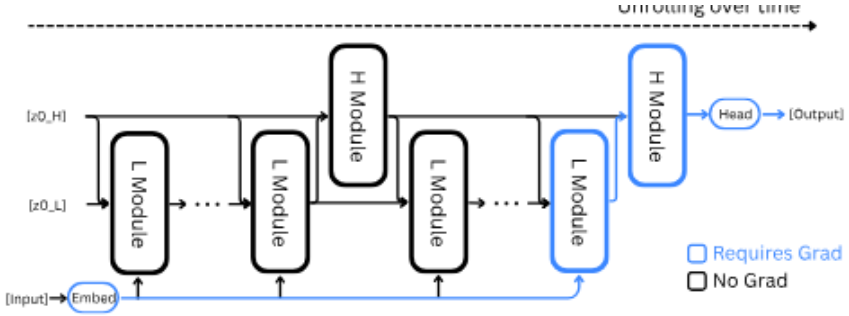
- Hierarchical Reasoning Model (Sapient Intelligence, arXiv 25.06)

```python
def hrm(z, x, n=2, T=2): # hierarchical reasoning
    zH, zL = z
    with torch.no_grad():
        for i in range(nT - 2):
            zL = L_net(zL, zH, x)
            if (i + 1) % T == 0:
                zH = H_net(zH, zL)
    # 1-step grad
    zL = L_net(zL, zH, x)
    zH = H_net(zH, zL)
    return (zH, zL), output_head(zH), Q_head(zH)

def ACT_halt(q, y_hat, y_true):
    target_halt = (y_hat == y_true)
    loss = 0.5*binary_cross_entropy(q[0], target_halt)
    return loss

def ACT_continue(q, last_step):
    if last_step:
        target_continue = sigmoid(q[0])
    else:
        target_continue = sigmoid(max(q[0], q[1]))
    loss = 0.5*binary_cross_entropy(q[1], target_continue)
    return loss

# Deep Supervision
for x_input, y_true in train_dataloader:
    z = z_init
    for step in range(N_sup): # deep supervision
        x = input_embedding(x_input)
        z, y_pred, q = hrm(z, x)
        loss = softmax_cross_entropy(y_pred, y_true)
        # Adaptive computational time (ACT) using Q-learning
        loss += ACT_halt(q, y_pred, y_true)
        _, _, q_next = hrm(z, x) # extra forward pass
        loss += ACT_continue(q_next, step == N_sup - 1)
        z = z.detach()
        loss.backward()
        opt.step()
        opt.zero_grad()
        if q[0] > q[1]: # early-stopping
            break
```

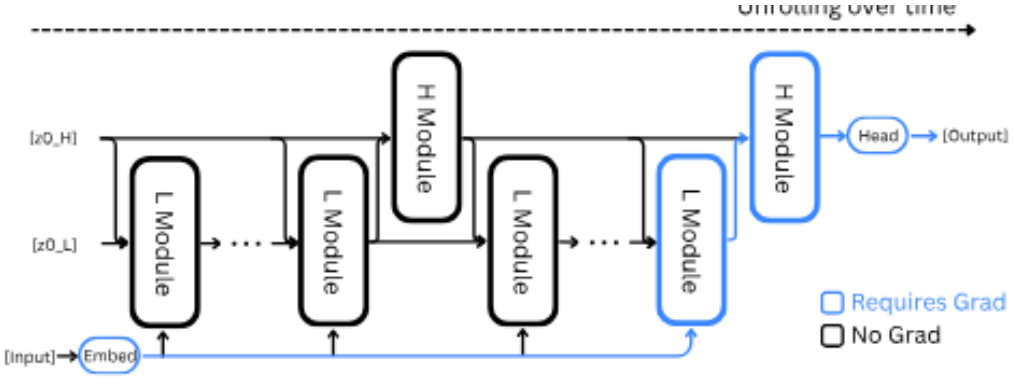


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

- Hierarchical Reasoning Model (Sapient Intelligence, arXiv 25.06)

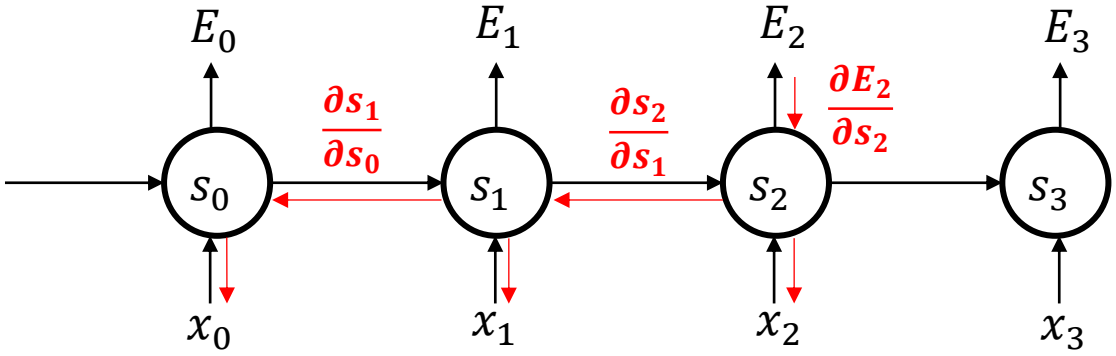
```
def hrm(z, x, n=2, T=2): # hierarchical reasoning
    zH, zL = z
    with torch.no_grad():
        for i in range(nT - 2):
            zL = L_net(zL, zH, x)
            if (i + 1) % T == 0:
                zH = H_net(zH, zL)
    # 1-step grad
    zL = L_net(zL, zH, x)
    zH = H_net(zH, zL)
    return (zH, zL), output_head(zH), Q_head(zH)
```



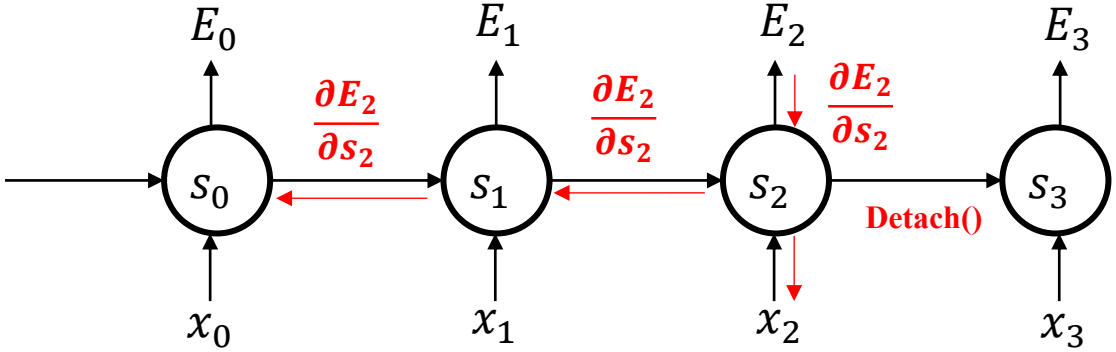
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

- Hierarchical Reasoning Model (Sapient Intelligence, arXiv 25.06)



BPTT(Backpropagation Through Time) (Werbos, 1990)



1-step Approximation (Geng, 2021)

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

- **Deep Equilibrium Model(DEQ): Ideal Segment State of HRM**
  - The low-level state z_L^* is defined as the **fixed point** of repeated updates**:**

$$z_L^* = f_L(z_L^*, z_H^{k-1}, \tilde{x}; \theta_L)$$

  - Where the iterative application of f_L cause the system to converge to the fixed point z_L^***.** When the **H-module** receives this converged result and performs a single update, it is defined as:

$$z_H^k = f_H\left(z_H^{k-1}, z_L^*, \tilde{x}; \theta_H\right), \qquad z_H^* = F\left(z_H^{k-1}, \tilde{x}; \theta\right)$$

  - Let $J_F = \frac{\partial F}{\partial z_H}$, At the fixed point, the gradient with respect to parameters satisfies:

$$\frac{\partial z_H^*}{\partial \theta} = J_F \frac{\partial z_H^*}{\partial \theta} + \frac{\partial F}{\partial \theta} \text{ which leads to: } \frac{\partial z_H^*}{\partial \theta} = (I - J_F)^{-1} \frac{\partial F}{\partial \theta}$$

  - Using the Neumann series expansion,

$$(I - J_F)^{-1} = 1 + J_F + J_F^2 + J_F^3 \ldots, (I - J_F)^{-1} \cong I$$

$$\frac{\partial z_H^*}{\partial \theta_H} \approx \frac{\partial f_H}{\partial \theta_H}, \quad \frac{\partial z_H^*}{\partial \theta_L} \approx \frac{\partial f_H}{\partial z_L^*} \cdot \frac{\partial z_L^*}{\partial \theta_L}, \quad \frac{\partial z_H^*}{\partial \theta_I} \approx \frac{\partial f_H}{\partial z_L^*} \cdot \frac{\partial z_L^*}{\partial \theta_I} \cdot \tag{2}$$

$$\frac{\partial z_L^*}{\partial \theta_L} \approx \frac{\partial f_L}{\partial \theta_L}, \quad \frac{\partial z_L^*}{\partial \theta_I} \approx \frac{\partial f_L}{\partial \theta_I} \cdot$$

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

- **Deep Supervision**
  - When the recursive structure of HRM is trained with **deep supervision**:
  - **1. Continuous feedback enables iterative updates,** similar to diffusion models that learn from the error at each step.
  - **2. Long-term credit assignment (e.g., the vanishing gradient problem)** can be handled approximately and locally, leading to more stable training.

```
# Deep Supervision
for x, y_true in train_dataloader:
    z = z_init
    for step in range(N_supervision):
        z, y_hat = hrm(z, x)

        loss = softmax_cross_entropy(y_hat, y_true)
        z = z.detach()

        loss.backward()
        opt.step()
        opt.zero_grad()
```


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

- **Adaptive computational time (ACT)**
  - The objective of **ACT** is to enable the model to **autonomously decide how much computation is required for each input.**
  - **Q-HEAD**
    - At segment m, the H-module produces Q-values via a Q-head:

$$\hat{\boldsymbol{Q}} = \boldsymbol{\sigma}\left(\boldsymbol{\theta}_{\boldsymbol{Q}}^{\boldsymbol{T}} \boldsymbol{z}_{\boldsymbol{H}}^{\boldsymbol{mNT}}\right) = (\hat{\boldsymbol{Q}}_{\boldsymbol{halt}}^{\boldsymbol{m}}, \hat{\boldsymbol{Q}}_{\boldsymbol{continue}}^{\boldsymbol{m}})$$

  - **Halting Policy**
    - The halting policy is constrained by a maximum and minimum number of segments, $\boldsymbol{M_{max}}$ **and** $\boldsymbol{M_{min}}$**.** With probability $\boldsymbol{\varepsilon}$**,** the minimum number of segments is sampled as $\boldsymbol{M_{min}} \sim \boldsymbol{Uniform}\{\mathbf{2}, \ldots, \boldsymbol{M_{max}}\}$ **,** and with probability $1 - \varepsilon$,$\boldsymbol{M_{min}} = \mathbf{1}$.
    - The model halts when:
    - $\boldsymbol{m} > \boldsymbol{M_{min}}$ && $\hat{\boldsymbol{Q}}_{\boldsymbol{halt}}^{\boldsymbol{m}} > \hat{\boldsymbol{Q}}_{\boldsymbol{continue}}^{\boldsymbol{m}}$ **OR** $\boldsymbol{m} > \boldsymbol{M_{max}}$
  - **ACT as Episodic MDP**
    - **State** at step m: z^m**, Action space** : {halt, continue}
    - The target values are defined as:

$$\hat{\boldsymbol{G}}^{\boldsymbol{m}} = (\hat{\boldsymbol{G}}_{\boldsymbol{halt}}^{\boldsymbol{m}}, \hat{\boldsymbol{G}}_{\boldsymbol{continue}}^{\boldsymbol{m}})$$

$$\hat{\boldsymbol{G}}_{\boldsymbol{halt}}^{\boldsymbol{m}} = \mathbf{1}\{\hat{\boldsymbol{y}}^{\boldsymbol{m}} = \boldsymbol{y}\},$$

$$\hat{\boldsymbol{G}}_{\boldsymbol{continue}}^{\boldsymbol{m}} = \begin{cases} \hat{\boldsymbol{Q}}_{\boldsymbol{halt}}^{\boldsymbol{m}}, & \boldsymbol{if}\ \boldsymbol{m} \geq \boldsymbol{M_{max}} \\ \boldsymbol{max}(\hat{\boldsymbol{Q}}_{\boldsymbol{halt}}^{\boldsymbol{m}}, \hat{\boldsymbol{Q}}_{\boldsymbol{continue}}^{\boldsymbol{m}}), & \boldsymbol{otherwise}. \end{cases}$$

  - **ACT Objective**

$$\boldsymbol{L}_{\boldsymbol{ACT}}^{\boldsymbol{m}} = \boldsymbol{Loss}(\hat{\boldsymbol{y}}^{\boldsymbol{m}}, \boldsymbol{y}) + \boldsymbol{BCE}(\hat{\boldsymbol{Q}}^{\boldsymbol{m}}, \hat{\boldsymbol{G}}^{\boldsymbol{m}})$$

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

- **Adaptive computational time (ACT)**

```
def ACT_halt(q, y_hat, y_true):
    target_halt = (y_hat == y_true)
    loss = 0.5*binary_cross_entropy(q[0], target_halt)
    return loss

def ACT_continue(q, last_step):
    if last_step:
        target_continue = sigmoid(q[0])
    else:
        target_continue = sigmoid(max(q[0], q[1])))
    loss = 0.5*binary_cross_entropy(q[1], target_continue)
    return loss
```

```
# Deep Supervision
for x_input, y_true in train_dataloader:
    z = z_init
    for step in range(N_sup): # deep supervision
        x = input_embedding(x_input)
        z, y_pred, q = hrm(z, x)
        loss = softmax_cross_entropy(y_pred, y_true)
        # Adaptive computational time (ACT) using Q-learning
        loss += ACT_halt(q, y_pred, y_true)
        _, _, q_next = hrm(z, x) # extra forward pass
        loss += ACT_continue(q_next, step == N_sup - 1)
        z = z.detach()
        loss.backward()
        opt.step()
        opt.zero_grad()
        if q[0] > q[1]: # early-stopping
            break
```


Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

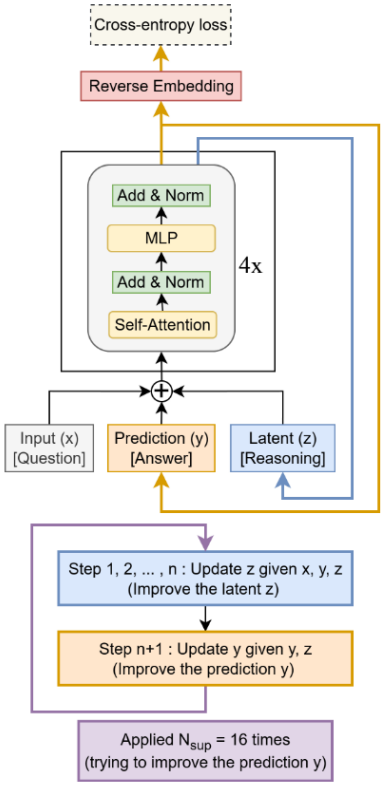
- Less is More: Recursive Reasoning with Tiny Networks (Samsung, arXiv, 2510)
  - 1. Limitations of the **fixed-point** assumption, particularly the reliance on one-step approximation.
  - 2. The H-module and L-module architecture contributes only **marginally** to performance.
  - 3. **Deep supervision** is the primary driver of strong performance.
  - 4. Q-learning can be replaced with a **simple binary classifier**.

```
def latent_recursion(x, y, z, n=6):
    for i in range(n): # latent reasoning
        z = net(x, y, z)
    y = net(y, z) # refine output answer
    return y, z

def deep_recursion(x, y, z, n=6, T=3):
    # recursing T-1 times to improve y and z (no gradients needed)
    with torch.no_grad():
        for j in range(T-1):
            y, z = latent_recursion(x, y, z, n)
    # recursing once to improve y and z
    y, z = latent_recursion(x, y, z, n)
    return (y.detach(), z.detach()), output_head(y), Q_head(y)

# Deep Supervision
for x_input, y_true in train_dataloader:
    y, z = y_init, z_init
    for step in range(N_supervision):
        x = input_embedding(x_input)
        (y, z), y_hat, q_hat = deep_recursion(x, y, z)
        loss = softmax_cross_entropy(y_hat, y_true)
        loss += binary_cross_entropy(q_hat, (y_hat == y_true))
        loss.backward()
        opt.step()
        opt.zero_grad()
        if q_hat > 0: # early-stopping
            break
```

Figure 3. Pseudocode of Tiny Recursion Models (TRMs).



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Hierarchical Reasoning Model

- Less is More: Recursive Reasoning with Tiny Networks (Samsung, arXiv, 2510)

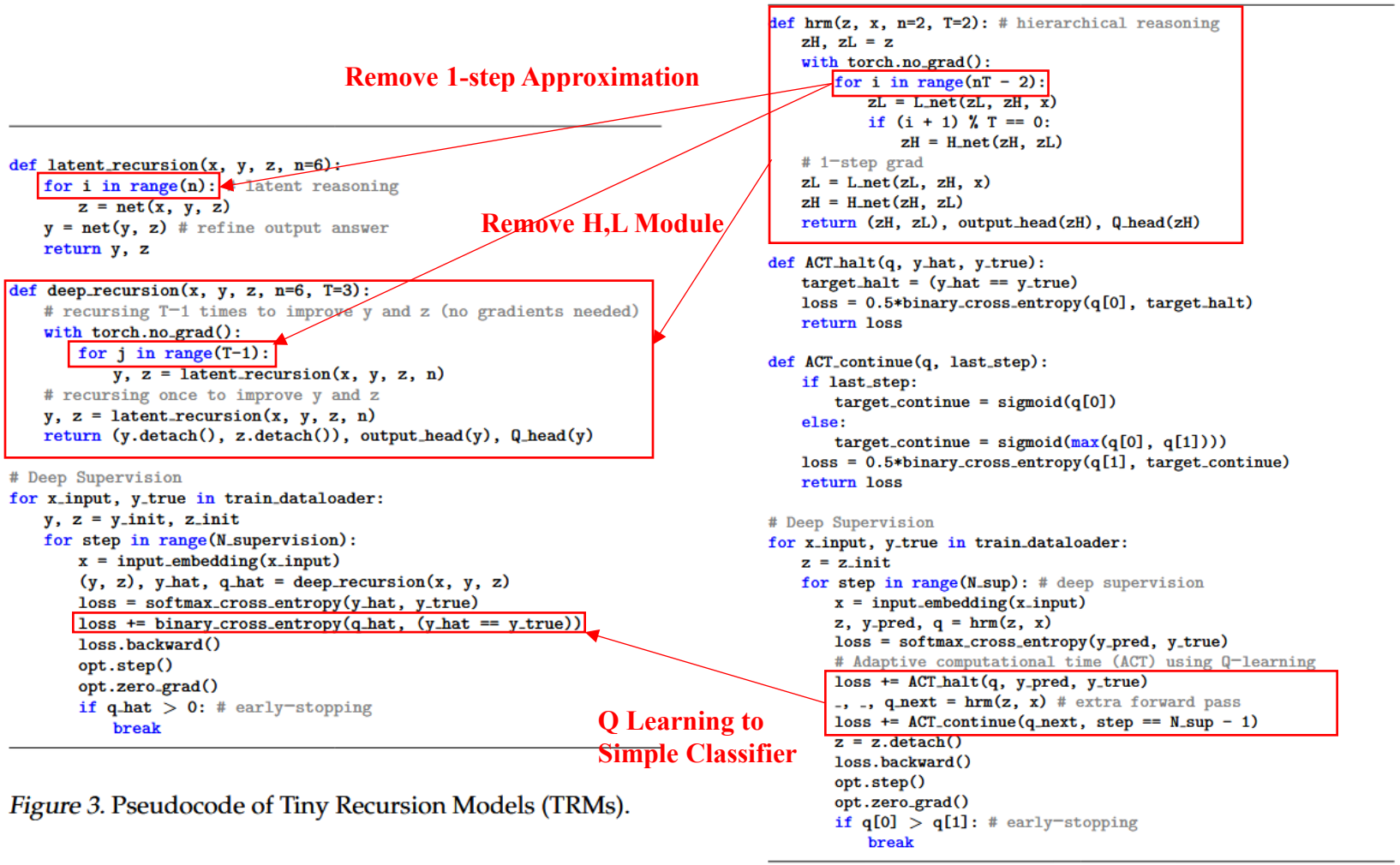
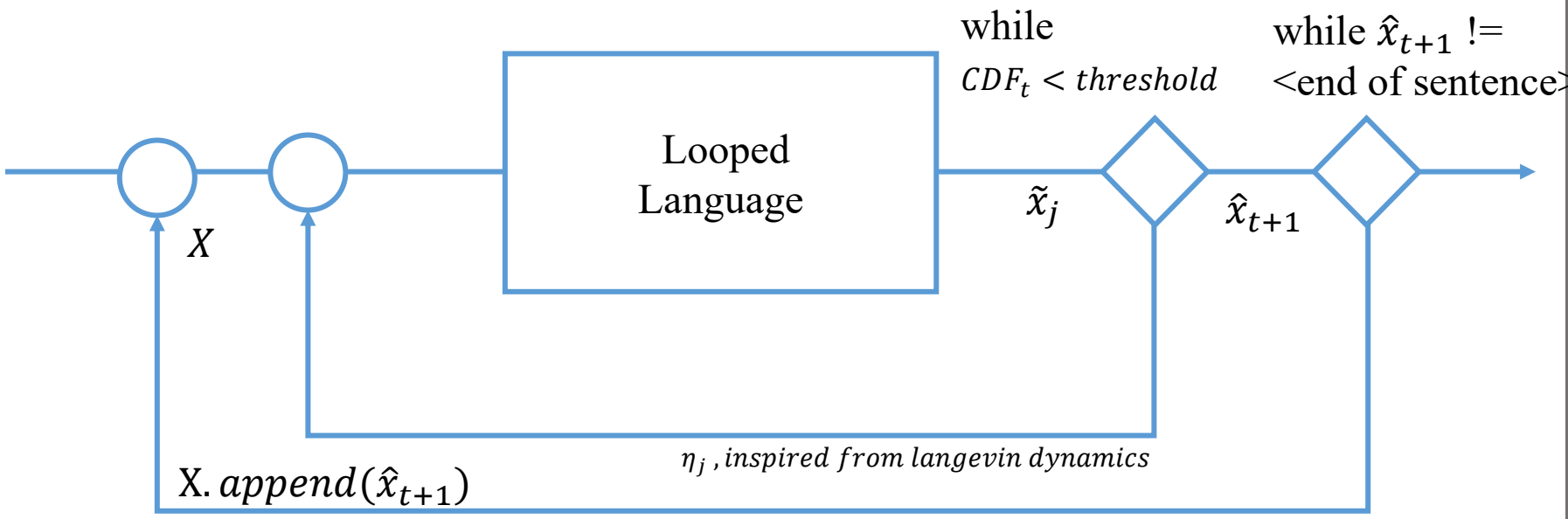


Figure 3. Pseudocode of Tiny Recursion Models (TRMs).

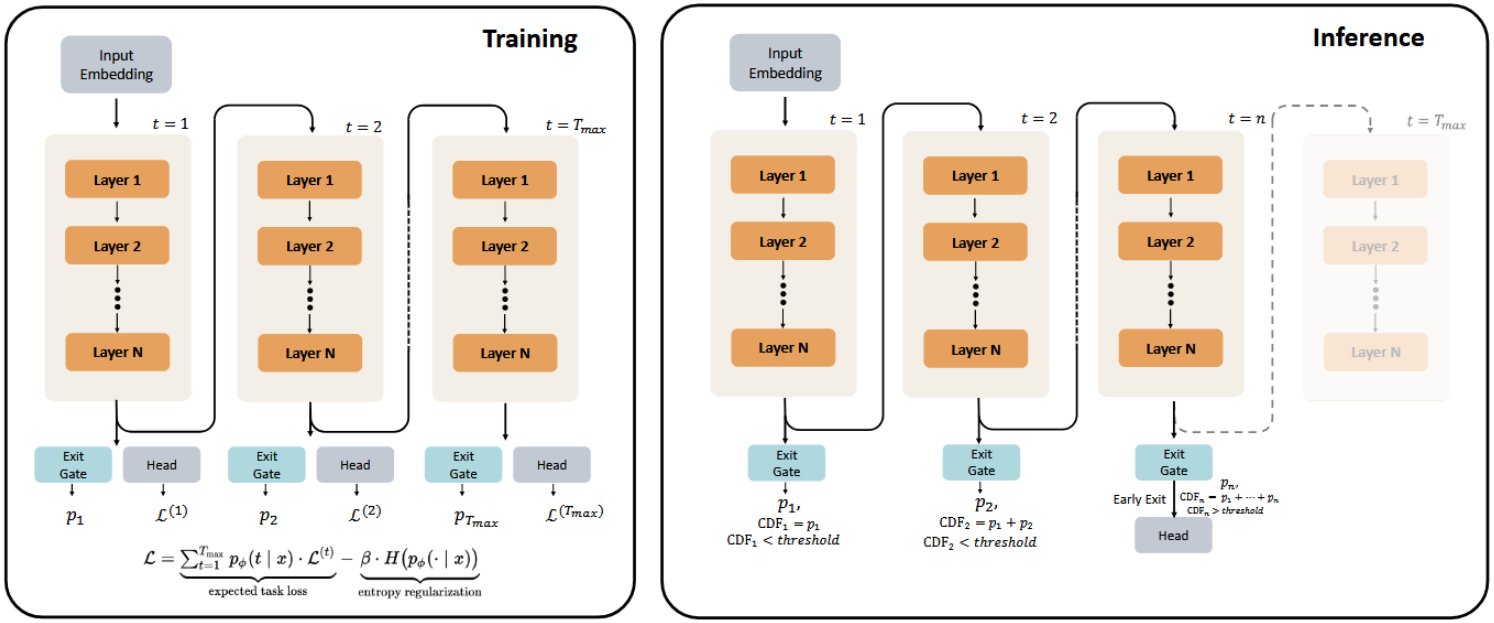
Scaling Latent Reasoning

- Scaling Latent Reasoning via Looped Language Models (ByteDance Seed, Zhu et al, arXiv, 25.10)



Scaling Latent Reasoning

- Scaling Latent Reasoning via Looped Language Models (ByteDance Seed, Zhu et al, arXiv, 25.10)



$$F(\cdot) := \text{lmhead} \circ \mathcal{M}^L \circ \text{emb}(\cdot), \qquad \mathcal{M}^L(\cdot) := \mathcal{T}_{\theta_L} \circ \cdots \circ \mathcal{T}_{\theta_1}(\cdot)$$

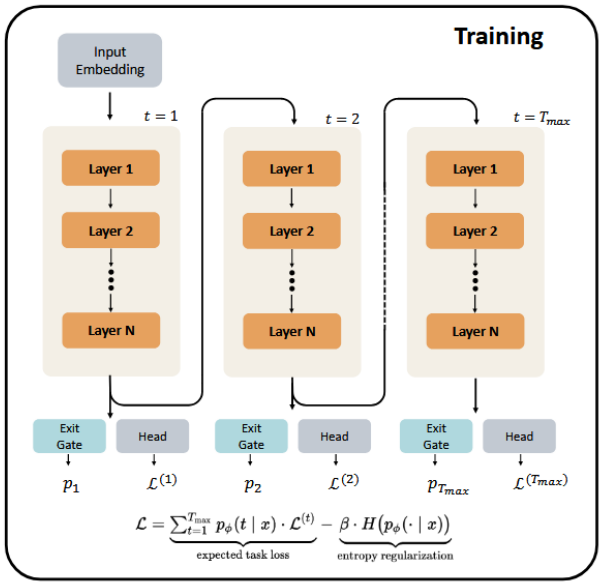
$$F^{(t)}(\cdot) = \text{lmhead} \circ \underbrace{\mathcal{M}^L \circ \mathcal{M}^L \circ \cdots \circ \mathcal{M}^L}_{t \text{ iterations}} \circ \text{emb}(\cdot). \tag{1}$$

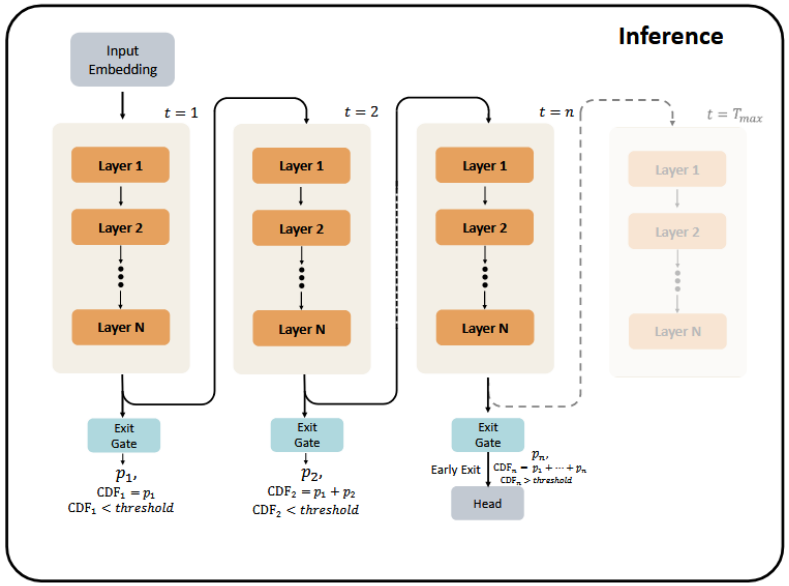
Scaling Latent Reasoning

- Scaling Latent Reasoning via Looped Language Models (ByteDance Seed, Zhu et al, arXiv, 25.10)
  - Exit probability: $\lambda_t(x) = \sigma\left(Linear_\phi\left(h^{(t)}\right)\right) \in (0,1)$
  - Not exiting in the first t steps : $S_t(x) = \prod_{j=1}^{t}(1-\lambda_j(x)), S_0(x) = 1$
  - $\widetilde{p_t}(x) = \lambda_t(x) S_{t-1}(x), t = 1, \ldots, T_{max} - 1$

$$p_\phi(t \mid x) = \begin{cases} \tilde{p}_t(x), & t = 1, \ldots, T_{\max} - 1, \\ S_{T_{\max}-1}(x), & t = T_{\max}, \end{cases} \quad \text{so that} \quad \sum_{t=1}^{T_{\max}} p_\phi(t \mid x) = 1. \qquad (3)$$

$$\mathrm{CDF}(n \mid x) = \sum_{t=1}^{n} p_\phi(t \mid x) = 1 - \prod_{j=1}^{n}(1 - \lambda_j(x)), \quad n < T_{\max}, \qquad \mathrm{CDF}(T_{\max} \mid x) = 1.$$





Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Scaling Latent Reasoning

- Scaling Latent Reasoning via Looped Language Models (ByteDance Seed, Zhu et al, arXiv, 25.10)
  - **Stable Training** (Stage 1a, Stage 1b) : Joint training of the LM and the gating mechanism for stable adaptive computation.
  - **CT Annealing** (Stage 2) : Freeze the LM and train only the gate to refine depth allocation.
  - **LongCT** (Stage 3) : Extend the context window using long-sequence training.
  - **Mid-training** (Stage 4) : Improve overall model capability using high-quality curated datasets.
  - **Reasoning SFT** (Post Training): Enhance reasoning performance through supervised fine-tuning on reasoning-focused data.

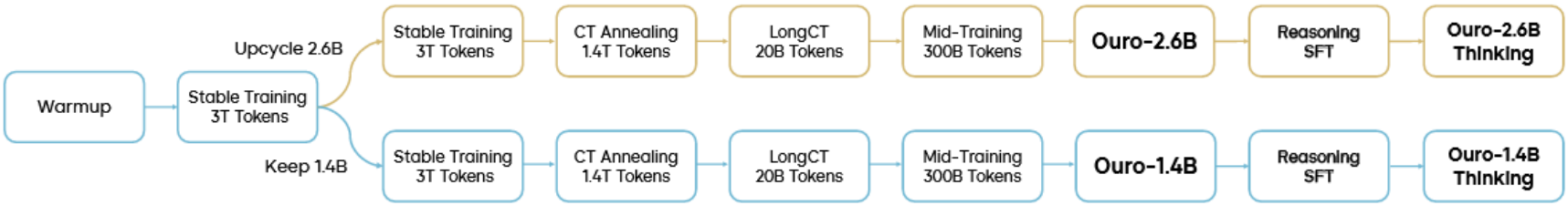
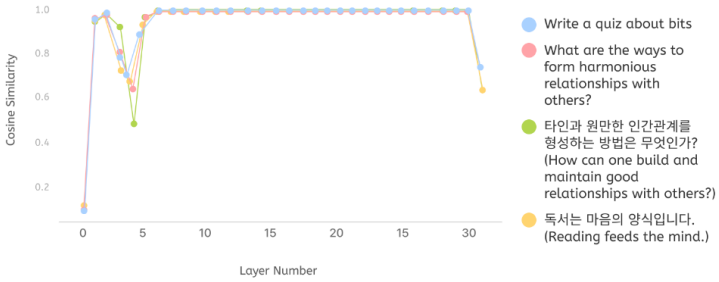


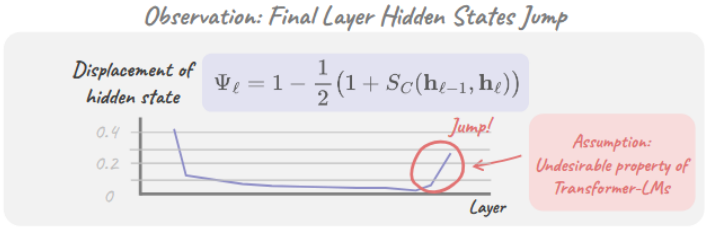
Figure 4 End-to-end Ouro training pipeline: shared warmup → Stable Training → forks into a 1.4B retained path and a 2.6B upcycled path → four shared stages → Reasoning SFT to produce Ouro-Thinking.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Deep Up-Scaling

From an analysis of the roles of Transformer blocks in LLMs**,** the early blocks perform representation in a latent space **(Enrichment Phase),** the middle blocks perform thinking **(Plateau Phase),** and the later blocks change the representation back to the original space **(Extraction Phase)** (Tikhonov et al., CIKM 2025).





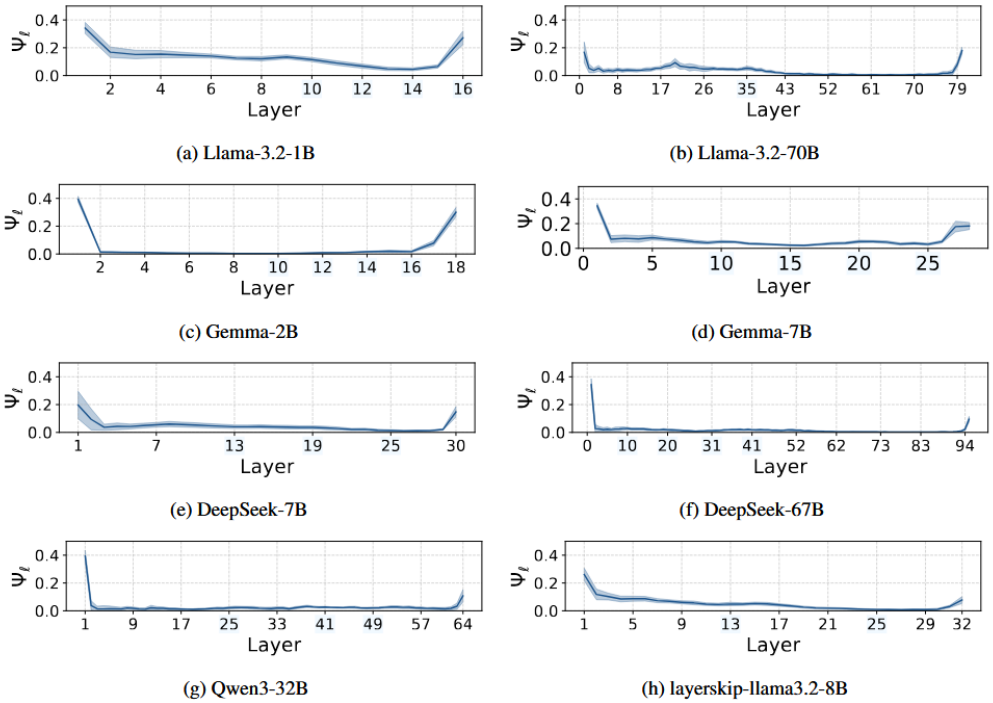
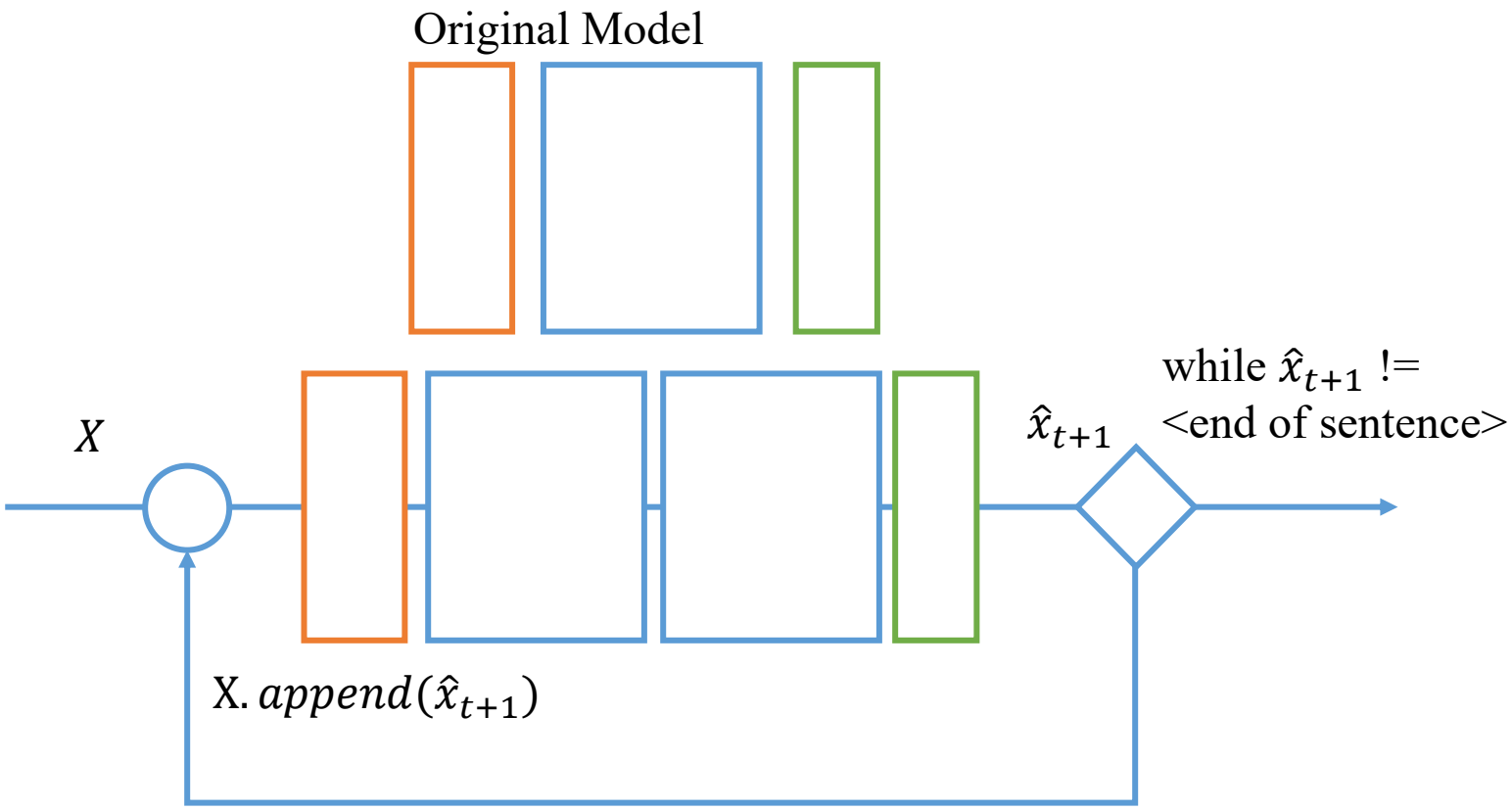


Figure 2: Layer-wise hidden state displacement Ψ_ℓ for next-word prediction on 100 samples from the LAMBADA dataset. Across all model architectures, the displacement at the final layer tends to be larger than that of the middle layers.

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Deep Up-Scaling

- **Upstage** (Upstage, NAACL 2024) achieved performance improvements by applying **Deep Up-Scaling** to the LLaMA 2.0 model. Similarly, **Mi:dm 2.0** (KT, arXiv 2601) applies **Deep Up-Scaling** to its internally pretrained model, which is submitted to **독파모**.



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Deep Up-Scaling

- **Deep Up-Scaling**

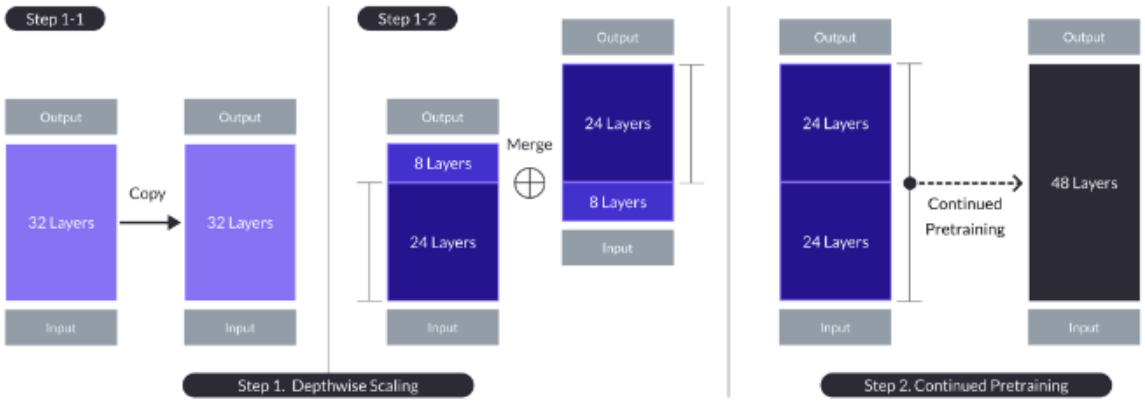
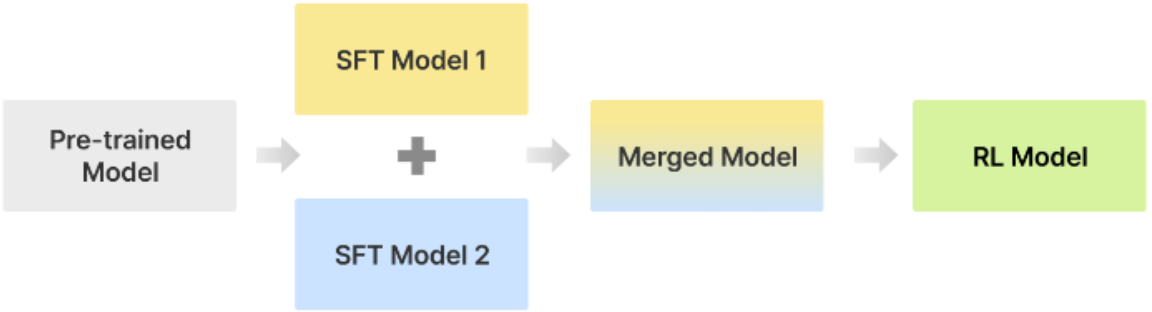


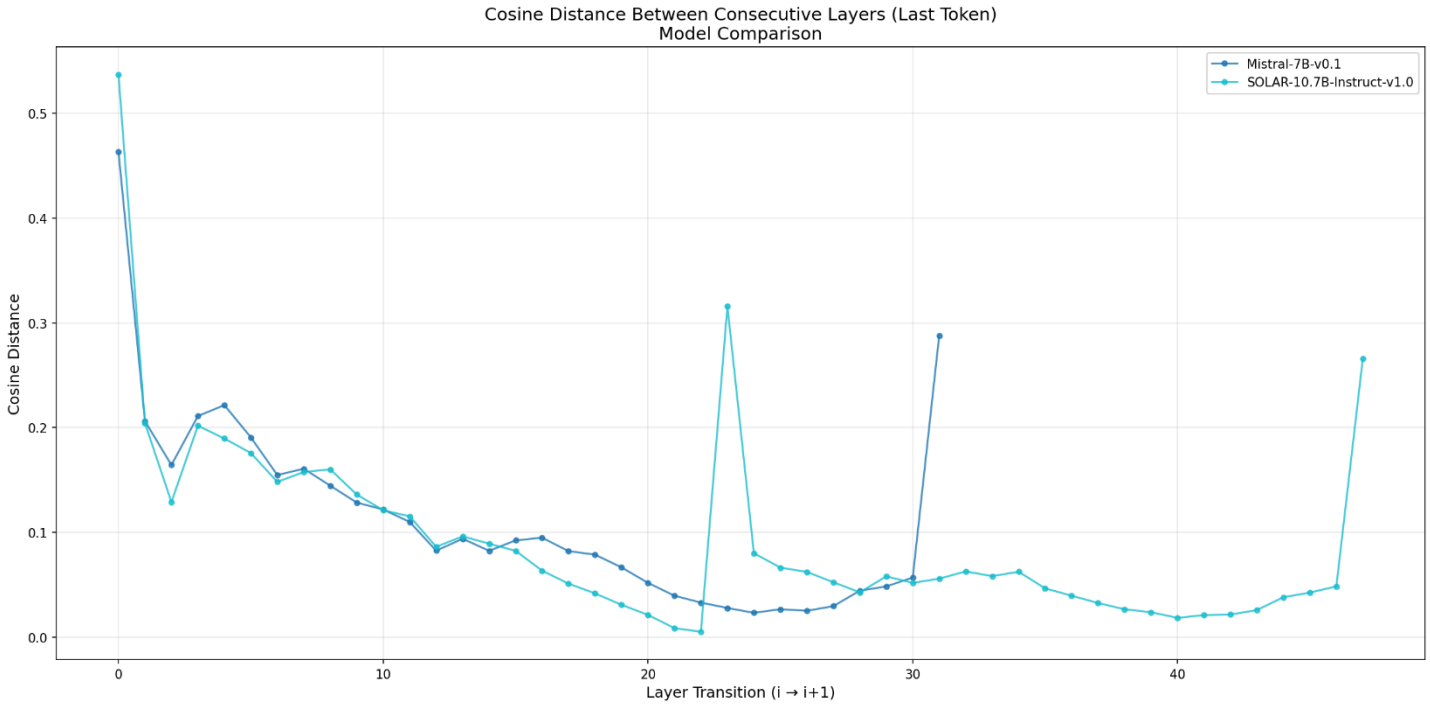
Figure 1: Depth up-scaling for the case with $n = 32$, $s = 48$, and $m = 8$. Depth up-scaling is achieved through a dual-stage process of depthwise scaling followed by continued pretraining.



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Deep Up-Scaling

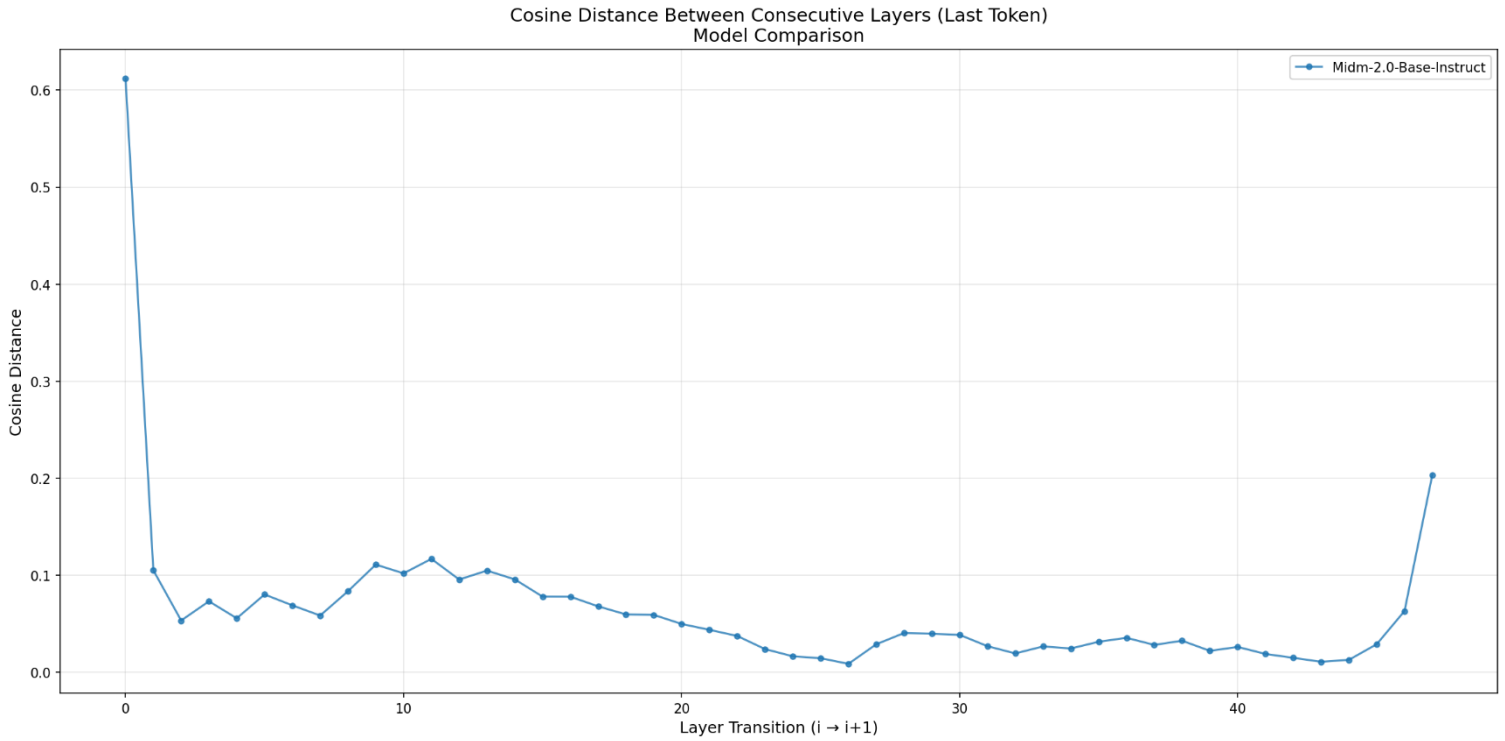
- **Deep Up-Scaling**



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

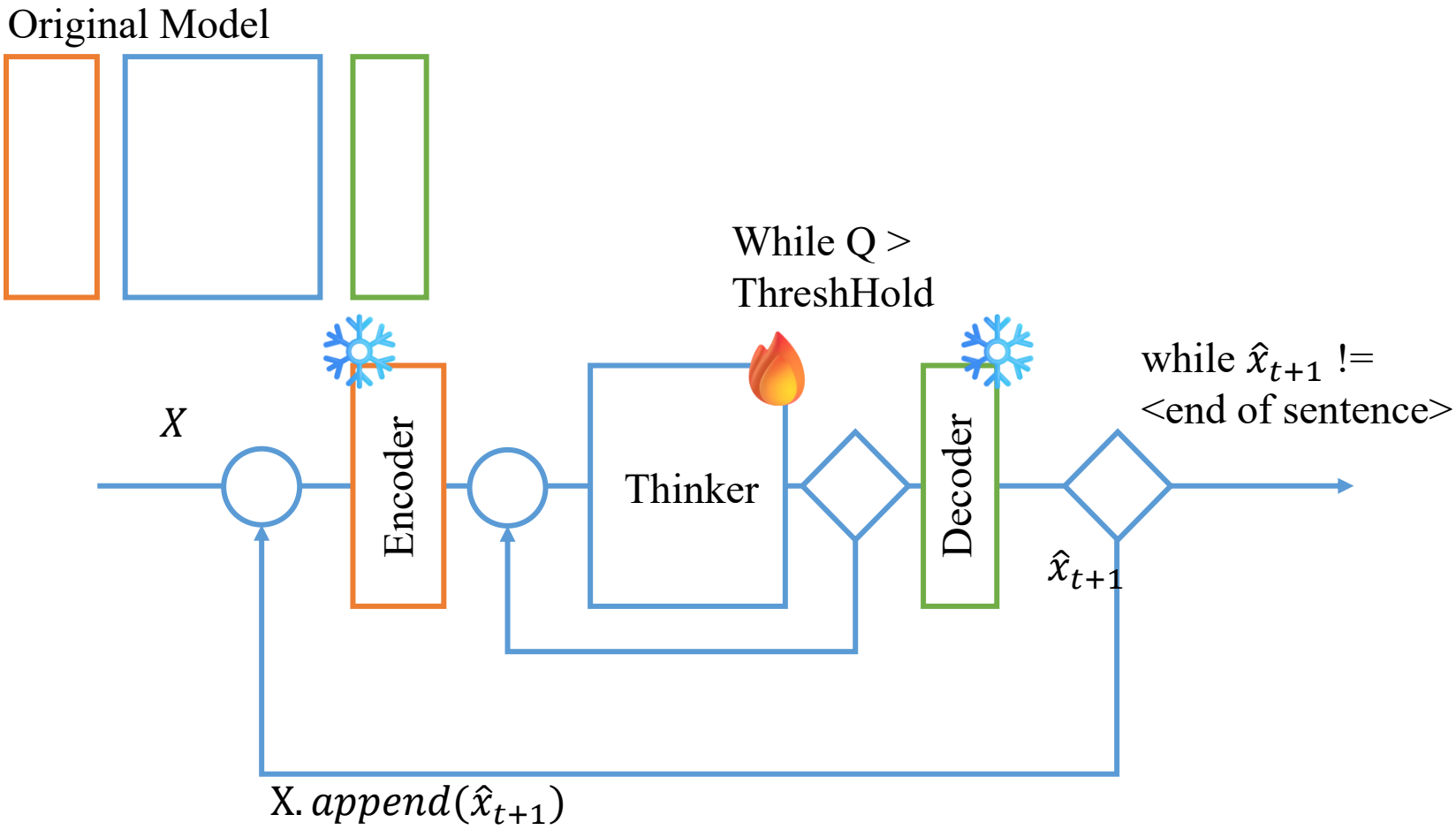
Deep Up-Scaling

- **Deep Up-Scaling**



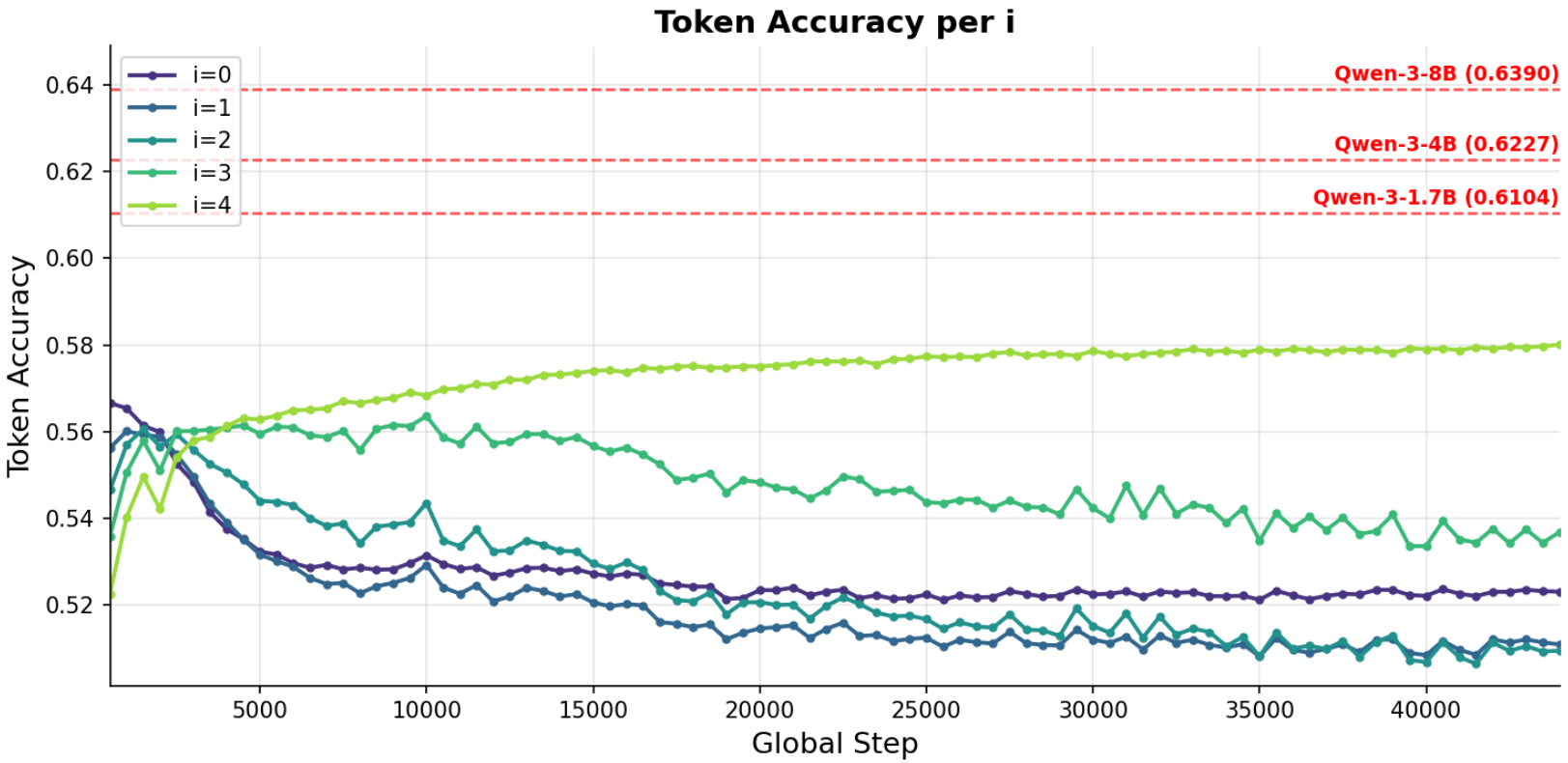
Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Future Work



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Future Work



Prof. Hyerim Bae (hrbae@pusan.ac.kr)

References

[1] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[2] Liu, Jian-wei, Bing-rong Xu, and Zhi-yan Song. "A Survey of Recursive and Recurrent Neural Networks." *arXiv preprint arXiv:2510.17867* (2025).

[3] Kahneman, Daniel. *Thinking, fast and slow*. macmillan, 2011.

[4] Snell, Charlie Victor, et al. "Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning." *The Thirteenth International Conference on Learning Representations*. 2025.

[5] Hoffmann, Jordan, et al. "Training compute-optimal large language models." *arXiv preprint arXiv:2203.15556* (2022).

[6] Leviathan, Yaniv, Matan Kalman, and Yossi Matias. "Fast inference from transformers via speculative decoding." *International Conference on Machine Learning*. PMLR, 2023.

[7] Wang, Guan, et al. "Hierarchical Reasoning Model." *arXiv preprint arXiv:2506.21734* (2025).

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

[9] Shao, Zhihong, et al. "Deepseekmath: Pushing the limits of mathematical reasoning in open language models." *arXiv preprint arXiv:2402.03300* (2024).

[10] Peebles, William, and Saining Xie. "Scalable diffusion models with transformers." *Proceedings of the IEEE/CVF international conference on computer vision*. 2023.

[11] Nie, Shen, et al. *"Large Language Diffusion Models."* Proceedings of the Thirty-Ninth Annual Conference on Neural Information Processing Systems, 2025

[12] Gladstone, Alexi, et al. *Energy-Based Transformers Are Scalable Learners and Thinkers*. Proceedings of the International Conference on Learning Representations (ICLR 2026), 2026. Oral Presentation.

[13] Geng, Zhengyang, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. "On Training Implicit Models." *Advances in Neural Information Processing Systems*, 2021.

[14] **Werbos, Paul J.** "Backpropagation Through Time: What It Does and How to Do It." *Proceedings of the IEEE*, vol. 78, no. 10, 1990, pp. 1550–1560. **IEEE**, https://doi.org/10.1109/5.58337.

[15] Jolicoeur-Martineau, Alexia.(Samsung) "Less is more: Recursive reasoning with tiny networks." *arXiv preprint arXiv:2510.04871* (2025).

[16] Tikhonov, Pavel, and Dmitry Ilvovsky. "Frozen in the Middle: Hidden States Remain Unchanged Across Intermediate Layers of Language Models." *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025.

[17] Kim, Sanghoon, et al(Upstage). "**SOLAR 10.7B: Scaling Large Language Models with Simple yet Effective Depth Up-Scaling**." *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 6,

[18] Shin, Donghoon(KT), et al. "Mi: dm 2.0 Korea-centric Bilingual Language Models." *arXiv preprint arXiv:2601.09066* (2026).

[19] Geiping, Jonas, et al. "Scaling up test-time compute with latent reasoning: A recurrent depth approach." *arXiv preprint arXiv:2502.05171* (2025).

Prof. Hyerim Bae (hrbae@pusan.ac.kr)

Q&A

Prof. Hyerim Bae (hrbae@pusan.ac.kr)