BRILLM-METAPRED: A UNIFIED-STATE LANGUAGE MODEL WITH ADAPTIVE SELF-MEASUREMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We present the **BriLLM-MetaPred** model, the first engineering realization of the *MetaPred* machine learning framework prescribed by an AGI mathematical theory, instantiated upon the task of language modeling. That theory argues MetaPred to be both necessary and sufficient as a machine learning primitive for general cognition. Taking its conclusion as our design premise, we report the engineering realization of this primitive for language modeling: it employs one ever-evolving dynamical state to serve at once as the context representation and as the carrier from which prediction is read, and by continual measurement of its own state it lowers finite-depth inconsistency, keeps inner coherence, and furnishes an endogenous, fixed-point form of dynamical self-supervision. The single dynamical state of MetaPred lets training and inference walk together of their own accord. Hence MetaPred, as a machine learning paradigm, shows symmetry across nearly every key element of machine learning—a trait that is but the plain and visible mark of the mathematical optimum within the space of all paradigms. BriLLM-MetaPred is the concrete bridge between two prior gifts: the MetaPred primitive's constructive elements and motive furnished by AGI theory, and the SiFu learning paradigm offered by BriLLM (Zhao et al., 2025)—whose node-signal-flow design lets that primitive run upon such software and hardware accelerators as are today at hand. Our realized BriLLM-MetaPred attains predictive performance comparable to a GPT baseline of equal size, giving first proof of the engineering feasibility of the MetaPred framework. MetaPred is the first machine learning paradigm to be derived in full from mathematical theory, not to be pieced together by human craft and improved inch by inch—and perchance the last of its kind.

1 METAPRED AS THE MACHINE LEARNING PARADIGM FOR AGI

The AGI mathematical theory (Anonymous, 2026) establishes a single machine learning primitive, MetaPred, as the necessary and sufficient condition for general cognition: the representations it forms and the predictions it emits are two sides of one and the same persistent dynamical state, which, under an endogenous fixed-point supervision, lowers its own finite-depth inconsistency through self-measurement, and so keeps inner coherence. We report the first engineering realization of this primitive, BriLLM-MetaPred, built upon the SiFu learning paradigm of BriLLM (Zhao et al., 2025)—SiFu supplies the node-signal-flow carrier that lets MetaPred run on existing accelerators; this paper supplies the objective function at the primitive level and a finite-budget self-measurement mechanism, so that the carrier squares with the MetaPred canon. Hence BriLLM-MetaPred is the executable bridge between AGI theory and BriLLM—no rival to Transformer language models in the usual architectural sense, nor a mere increment upon BriLLM.

To be plain: the model is a node-state language model with a single recurrent tensor. Token projection vectors are written into fixed interface rows; a recurrent operator, evolving by its own dynamics, updates the whole tensor; and the next-token distribution is read from chosen output rows and a bounded auxiliary pool. One and the same evolving tensor bears at once the input-conditioned representation and the state from which reading is made; the output projection is but a parameterized map, its input drawn from that same evolving state. MetaPred holds an endogenous fixed-point prediction aim; external labels curb only the emitted distribution, not the track of the inner state. Its

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

self-measuring mechanism starts from the same input condition, compares the successor state of one evolution with a deeper simulated state, and turns the difference to train the shared dynamics, to supervise the direct-measurement process, and to afford a bounded course for state correction.

The Two Constituents of a Machine Learning Paradigm Following Zhao et al. (2025), any machine learning paradigm may be defined by how it represents semantic objects and how it executes prediction. Mainstream paradigms encode meaning into vectors and transform them by learned functions. The SiFu learning paradigm of BriLLM (Zhao et al., 2025) does otherwise: it maps semantic units onto dedicated nodes, and prediction emerges through signal flow among nodes. SiFu is a spatial, component-wise scheme of semantic representation, wherein distinct components directly bear distinct meanings as the user appoints; traditional deep learning is a temporal scheme, wherein one and the same vector shifts with time.

SiFu and BriLLM together break a tacit curb of existing paradigms: the deep binding of model carrier to dynamical state. Existing models commonly require architecture and dynamical state to align strictly in shape and topology, and to hold that alignment through every evolutionary step. SiFu breaches this from two sides: the model carrier is a fully bidirected, fully linked graph of the greatest generality; and BriLLM lets the dynamical state stand not wholly aligned with the model architecture—each training sample, in training and in inference, answers to a certain subset of the maximal dynamical state space. Once the alignment curb is loosed, the model may weigh time against space more freely, and this is the very means by which BriLLM bears unbounded input and output at constant model size.

The MetaPred Primitive The AGI mathematical theory (Anonymous, 2026) asks: if a system is to gain unbounded improvement in predictive performance and efficiency under no unbounded external supervision, what relation must it keep within? The answer, the theory argues, is MetaPred. This paper takes that conclusion as a design premise, and puts the primitive to the test upon a language task: is it runnable, is it measurable, and can the model so realized give results reasonable—nay, comparable to those of existing models?

The space of candidate paradigms weighed by the theory is just the general case of weak coupling between model carrier and dynamical state as SiFu defines it, wherein the dynamics may be told as a fiber bundle over a topological carrier. Since MetaPred is more general than existing machine learning in the paradigmatic sense, its engineering realization must rest upon the generalized topological structure that SiFu suggests.

MetaPred bears representation and prediction at once upon one single, ever-evolving dynamical state—this is its chief trait, setting it apart from every existing paradigm. Hence MetaPred shows symmetry across all core elements of machine learning: symmetry of representation and prediction, of training aim and endogenous supervision, of training process and inference process. These symmetries are the plain and visible marks of the mathematical optimum within the space of paradigms.

MetaPred is unlike any existing machine learning paradigm or model, even where they share such surface words as recurrence, self-measurement, attention, dynamics, representation, or prediction. BriLLM-MetaPred is a generative language model upon the SiFu carrier under the MetaPred frame, wholly other in paradigm from GPT (Radford et al., 2018) built on Transformer (Vaswani et al., 2017). Its backbone is drawn from the optimization of mathematical theory, and it holds a generality of carrier and of dynamical realization greater than the existing GPT paradigm.

Paradigmatic Difference from Hopfield and Kindred Mechanisms In a Hopfield network, dynamics serves to fetch stored memory: the fixed point is the memory content itself, convergence is guaranteed by design, and the final state is wholly fixed by the set of attractors; the modern Hopfield network is strictly equivalent to Transformer attention, its single-step update in the adiabatic approximation being the self-attention layer (Hopfield, 1982; Ramsauer et al., 2021). In MetaPred, dynamics itself is computation: each evolutionary step yields representation and prediction together; the fixed point is no preset target but a curb of self-consistency, and convergence is but a conditional result. Thus Hopfield's dynamics answers, "which memory does this cue match?", while MetaPred's dynamics answers, "how should the next step evolve?" The rift is paradigmatic, not architectural.

As to training aims, Hopfield reaches its minima by a monotone fall of a Lyapunov-style energy; MetaPred builds an endogenous supervision by self-measurement, and realizes self-supervised

learning under a relative mean-squared error. The difference in the role of the fixed point springs from this very source: for the one it is the memory content, for the other a curb of self-consistency.

As to widening the degrees of freedom of dynamical evolution over the model carrier, several advances have been made along three courses—module coordination, sequence modeling, and deep recursion—yet all rest upon a certain architecture. Goyal et al. (2022), moved by the global workspace theory, advanced a "shared workspace" as a general module-coordination mechanism, validated on Transformer, and much akin to the self-coordination mechanism in MetaPred. Gu et al. (2022) proposed a sequence-modeling layer grounded in structured state space, putting a linear time-invariant system in place of the quadratic-complexity attention, whose dynamics thus stands free of input content. Dehghani et al. (2019) introduced the Universal Transformer, which realizes deep recursion through parameter sharing across layers, secures dynamic computation, and gains Turing completeness together with stronger compositional generalization. MetaPred, by contrast, realizes dynamical evolution upon a general SiFu topology; save for the external task aim, its own fixed-point supervision, and the SiFu topology, its dynamics stands free of any further architectural curb. It is therefore no patchwork of engineering tricks, but a whole end-to-end design from mathematical theory; and those partial designs find, in the AGI context, each its own rightful place.

Paradigmatic Link to the Working of the Brain To realize MetaPred one must choose a distance functional by which to tell state signals apart. Within the frame of global signal transforms, time-domain amplitude and frequency-domain phase form a twin pair of representations. This paper takes time-domain amplitude, in service of ease with existing computer hardware and software; the theory keeps open the concrete form of the distance functional. The brain, in fact, uses a frequency-domain means, telling states apart by oscillatory phase. That divergence is no paradigmatic breach.

Electrophysiological evidence shows that the phase-amplitude mechanism of brain oscillations is just how predictive coding is realized at the neuronal-population level: the phase of slower oscillations mirrors the excitability of postsynaptic neurons, thereby timing faster synaptic input; gamma oscillations encode bottom-up prediction errors, while beta oscillations take part in top-down prediction (Noda et al., 2017); cross-frequency coupling maps the hierarchical play of predictive coding onto certain frequency bands (Arnal & Giraud, 2012). Cortical network models further show that inference in predictive coding naturally gives rise to oscillatory dynamics (Bastos et al., 2012).

The self-measurement residual of MetaPred answers in structure to this prediction-error signal: a one-step simulation is prediction, a K-step simulation is a deeper inference, and the difference between the two is error. When external labels fail, this error signal still drives the dynamics toward inner consistency—in tune with the core tenet of predictive coding, that prediction-error minimization serves as the local learning signal, and in tune too with the free-energy account of the brain (Friston, 2010).

Therefore, setting aside the certain choice of distance functional, BriLLM-MetaPred stands in high accord with the brain in its working mechanism: it realizes representation and prediction together through one and the same dynamical evolution, and begets endogenous supervision from a finite-depth self-comparison. This accord is no design premise—BriLLM-MetaPred's design comes wholly from mathematical theory; its correspondence with known mechanisms of the brain is an independent observation worthy of serious further inquiry.

2 The BriLLM-MetaPred Model

This section tells the whole realization of the BriLLM-MetaPred model; its overall architecture appears in Figure 1.

2.1 Notation and Overall Architecture

BriLLM-MetaPred is realized as a node-graph neural network, its node signals forming a tensor $\mathbf{S} \in \mathbb{R}^{N \times d}$. The node set V is split into five mutually exclusive subsets: the input region I, the output region O, the self-coordination node E, the modal region M, and the working region W; and there are further defined the adaptive internal node region $U = M \cup W$, the coordination region $C = \{E\} \cup M$, and the read-out auxiliary region $F = I \cup W \cup O$. The graph is fully bidirected;

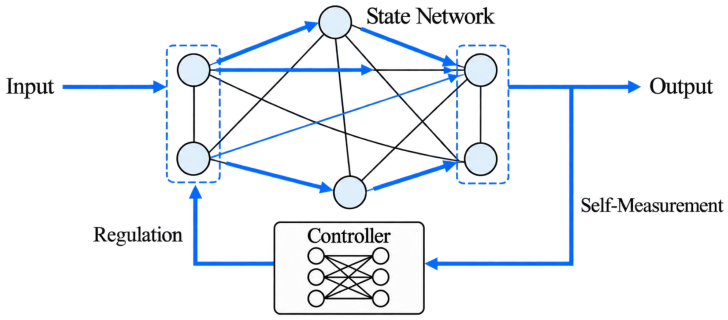

Figure 1: The architecture of MetaPred

the path from input to output passes through $M \cup W$ or E, so as to bar a short circuit from input to read-out; talk between nodes is done by three node-axis mixing operators.

The evolutionary state $\mathbf{S}_{t,u}$ is indexed by the external token step t and the internal evolutionary step u. The parameter-group index is $q_u = u \bmod Q$, cyclically reused. The learnable parameters fall, by time scale, into three families: slow parameters $\boldsymbol{\theta}$ (shared across sequences, batch-updated—projection, evolution, and prediction weights among them), fast parameters $\boldsymbol{\phi}_u$ (sequence-local, event-updated—fast prediction and correction parameters among them), and control parameters $\boldsymbol{\psi}$ (shared across sequences, updated by policy track, bound to the controller). The sequence runtime tuple $z_u = (\mathbf{S}_u, \boldsymbol{\phi}_u, \mathcal{H}_u, \mathcal{B}_u, q_u, \xi_u)$ keeps state, history, budget, and the random stream; it is reset at the sequence boundary, and only $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ are shared across sequences. The model holds two independent networks: the state network (a SiFu graph, bearing $\boldsymbol{\theta}, \boldsymbol{\phi}_u$) and the controller network (a feedforward policy-value network, bearing $\boldsymbol{\psi}$).

BriLLM-MetaPred holds two neural networks with free parameters. The state network, realized by the SiFu graph, bears the slow parameters $\boldsymbol{\theta}$ and the fast parameters $\boldsymbol{\phi}_u$, and is charged with the main state evolution; the controller network is a feedforward policy-value network, bearing the control parameters $\boldsymbol{\psi}$, and is charged with setting runtime conditions. The two interact through the state tensor and the runtime conditions.

2.2 The Two Elementary Dynamical Processes

MetaPred's dynamics contains only two elementary processes: main state evolution and self-measurement, sharing the standard dynamical routine Φ_q^{step}. Main state evolution runs upon the runtime state and moves u; self-measurement, under the controller's sway, recurses upon a copy of the state and leaves u as it was, making up a second-order recursive sub-process. Main state evolution takes two modes, with input (encoding / generation) and without input (autonomous). Prediction read-out moves not u, and is no part of evolution.

Single-step evolution $\Phi_q^{\text{step}}(\cdot; \chi) : \mathbb{R}^{N \times d} \to \mathbb{R}^{N \times d}$ comprises five gated-mixing stages run in order, where χ is the historical-dependency condition set holding the parameter-group index, the invocation stage, the input window content, and the remaining budget:

1. **Input mixing**: take in token information;
2. **Role assignment**: nodes lean of their own will among work / modal / fast / slow functions;
3. **Coordination aggregation**: squeeze the whole into a coordination anchor;
4. **Broadcast distribution**: make global information local;
5. **Gated writing**: write back to the state by fast and slow gates.

Continuous K-step composite is $\Phi_q^{(K)}$; true evolution moves u, simulated evolution leaves u as it was.

Self-measurement shares the same standard dynamical routine Φ_q^{step} with main state evolution; the difference is this: main state evolution runs this routine upon the runtime state and moves the evolutionary step u, whereas self-measurement, under the controller's sway, recurses this routine upon a copy of the same state and leaves the evolutionary step u as it was. Self-measurement is therefore a second-order recursive dynamical sub-process, nested above main state evolution, kept by the controller; it is not a static loss term standing apart from the dynamics.

Self-measurement builds two deterministic simulations:

$$\mathbf{S}_{\text{pred}} = \Phi_q^{\text{step}}(\mathbf{S}_u; \chi_u^{\text{meas}}), \qquad \mathbf{S}_{\text{sim}}^{(K)} = \Phi_q^{(K)}(\mathbf{S}_u; \chi_u^{\text{meas}}),$$

the residual $\Delta_{\text{self}}^{(K)}(u) = \mathbf{S}_{\text{sim}}^{(K)} - \mathbf{S}_{\text{pred}}$ measures finite-depth inconsistency. State correction reads

$$\mathbf{S}_{u+1} = \mathbf{S}_{u+1}^{\text{pre}} + \eta_{\text{eff}} \widetilde{\Delta}_{\text{self}}^{(K)}(u),$$

where $\mathbf{S}_{u+1}^{\text{pre}} = \Phi_q^{\text{step}}(\mathbf{S}_u; \chi_u^{\text{real}})$. The loss $\mathcal{L}_{\text{self}}$ is the sequence-level normalized square of the residual. Direct measurement predicts the one-step successor and the residual from parameters, and the loss $\mathcal{L}_{\text{direct-self}}$ is supervised by $K = 2$ simulation. Self-measurement runs within a conditional-convergence frame: when the full parameter-group cycle is a contraction, the Banach fixed-point theorem warrants a sole fixed point.

2.3 Working Architecture

External tokens are first looked up through the input projection matrix $E_{\text{in}} \in \mathbb{R}^{|\mathcal{V}| \times d}$, giving the token embedding vectors. Then a sliding-window mechanism updates the input representation and writes it into the input region I of the state tensor $\mathbf{S}$. This writing advances only the external time step t, and triggers encoding, autonomous, or generation evolution according to the current context. The initial state is set to zero and wholly reset at the sequence boundary.

The read-out layer draws pooled features o, i, c, f, m from the evolutionary state, builds a base feature r_{base}, and superposes a structural residual δ_{str} and a slot-alignment residual δ_{slot,k_j}, giving the full feature $r_j = r_{\text{base}} + \delta_{\text{str}} + \delta_{\text{slot},k_j}$. The final logits and probabilities read

$$\ell_j = W_{\text{out}} r_j + b_{\text{out}}, \qquad p_j = \text{softmax}(\ell_j),$$

and read-out moves not u.

The MetaPred model holds two neural networks with free parameters: the state network and the controller network. The controller network is a feedforward policy-value network, bearing the control parameters $\boldsymbol{\psi}$. It takes the causal state summary o_u as input and gives out a six-dimensional action a_u, whose dimensions rule respectively: the measurement tier, the measurement focus, the fast-parameter update switch, the update strength, the state-correction strength, and the sampling ratio. Upon a_u, the controller sets the runtime conditions χ_u, and thereby keeps the run of the second-order recursive dynamical sub-process (such as the scheduling and depth of self-measurement) under its unified control, rather than leaning on a fixed flow.

MetaPred has three running modes: Train (all parameters updated, self-measurement on), Frozen (parameters frozen, diagnostic only), and Adaptive (only fast parameters and correction live, to mock deployment). All three share one evolutionary skeleton, solely fixed by the single state and the endogenous supervision curb.

The split into three modes follows straight from two theoretical curbs: the single state needs that training and inference share one and the same evolutionary skeleton; endogenous supervision together with training–inference coupling needs that self-measurement be run in full under every mode. Evaluation metrics are computable under all three modes, yet are not used for parameter update in Frozen and Adaptive. The dynamical evolution of MetaPred is shown in Figure 2.

2.4 Training Algorithm

The three parameter families are updated apart by time scale: the slow parameters $\boldsymbol{\theta}$ by batch across sequences, the fast parameters $\boldsymbol{\phi}_u$ by event within a sequence, the control parameters $\boldsymbol{\psi}$ by policy track. The training aim reads

$$\mathcal{L}_{\text{outer}} = \mathcal{L}_{\text{task}} + \lambda_{\text{int}} \left(\lambda_p \mathcal{L}_{\text{pred}} + \lambda_f \mathcal{L}_{\text{fp}} + \lambda_d \mathcal{L}_{\text{dyn}} + \lambda_c \mathcal{L}_{\text{direct-self}} \right) + \lambda_s \mathcal{L}_{\text{self}},$$

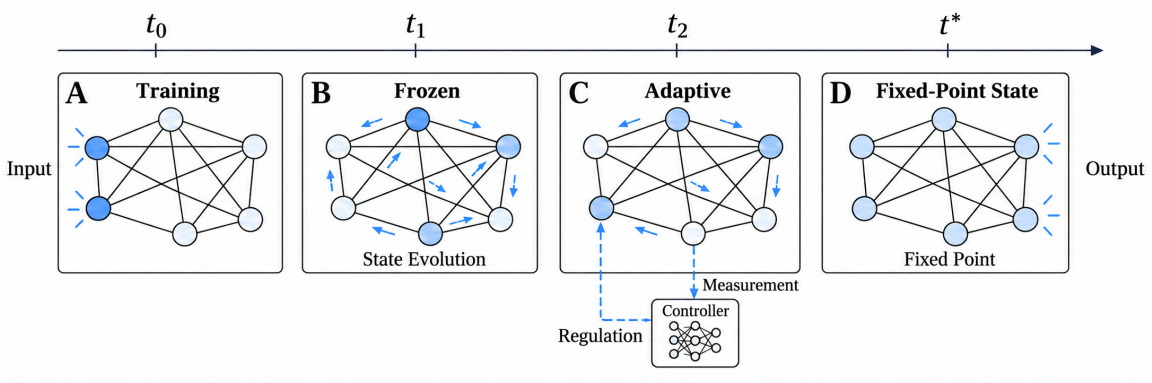

Figure 2: The dynamical evolution of MetaPred

Algorithm 1: The BriLLM-MetaPred training algorithm

Input: training corpus, budgets $K_{\text{in}}, K_{\text{ev}}$, learning rates $\alpha_\theta, \alpha_\phi$, PPO frequency T_{ppo}
Output: trained $\boldsymbol{\theta}, \boldsymbol{\phi}_u, \boldsymbol{\psi}$
Initialize $\boldsymbol{\theta}, \boldsymbol{\psi}, \boldsymbol{\phi}_u$.
for *each training batch* **do**
  for *each sequence* $x_{1:T}$ **do**
    Initialize runtime z_0.
    for $t = 1$ *to* T **do**
      Write e_t, run K_{in} encoding steps, store snapshot.
    Run K_{ev} autonomous steps.
    for $j = 1$ *to* H **do**
      (1) Controller picks action a_u.
      (2) Run measurement and fast-parameter update as asked.
      (3) Run true evolution to get $\mathbf{S}_{u+1}^{\text{pre}}$, apply correction as asked to get $\mathbf{S}_{u+1}$.
      (4) Read out prediction, compute $\mathcal{L}_{\text{task}}$, store transition observation $\mathcal{R}_i$. If $j < H$, write the next token.
    Accumulate gradients, update $\boldsymbol{\theta}$ by AdamW.
    if *PPO track reaches* T_{ppo} **then**
      Compute advantages by GAE, update $\boldsymbol{\psi}$ by PPO.

where $\mathcal{L}_{\text{task}}$ is the next-token cross-entropy; $\mathcal{L}_{\text{pred}}$ supervises the state predictors; $\mathcal{L}_{\text{fp}}$ probes the fixed point; $\mathcal{L}_{\text{dyn}}$ curbs trend consistency; $\mathcal{L}_{\text{direct-self}}$ calibrates the direct-measurement parameters; and $\mathcal{L}_{\text{self}}$ is the endogenous fixed-point self-supervision signal. The transition observation $\mathcal{R}_i$ is the common input to direct-measurement calibration, slow-update preview, and PPO.

Updates fall into two kinds: slow and fast parameters by direct gradient, control parameters by PPO policy gradient. Slow parameters are batch-updated by AdamW; fast parameters are updated by clipped SGD at the controller's ask; control parameters by delayed PPO update. The gradient paths stand apart, yet the state tensor $\mathbf{S}_{t,u}$ stays single, and training and inference share one and the same evolutionary skeleton. The whole flow appears in Algorithm 1.

3 Experiments

This section weighs the language-modeling performance of BriLLM-MetaPred. Two training schedules are used: the continuous-execution mode, which keeps node state and fast parameters across fragments within a sequence; and the sparse-auxiliary mode, which runs self-measurement now and then alongside the backbone training. Evaluation takes the Frozen and Adaptive modes, differing only in whether fast-parameter update and state correction are enabled; evaluation metrics are computable under both modes, yet are not used for parameter update.

Language modeling reports validation-set NLL, $\mathrm{PPL} = \exp(\mathrm{NLL})$, and $\mathrm{BPC} = \mathrm{NLL}/\ln 2$ for character tasks; lower is better in each case. Three seeds, 42, 43, 44, are used, with mean and sample standard deviation told.

English uses BookCorpusOpen: after whole-book dedup, a 98:1:1 split by document hash; the official GPT-1 BPE vocabulary of 40,478 entries is shared. Chinese uses THUCNews-v3: a 90:5:5 split by near-duplicate article group; the vocabulary is the 6,512 tokens that appear at least 5 times in the training set.

The GPT baseline is implemented strictly to the GPT-1 specification, scaled to the same size as BriLLM-MetaPred (about 13.2M parameters), and shares the data split and vocabulary; the difference lies in model architecture and training aim. The 129M-level baseline follows the official GPT-1 parameter scale. No 129M equal-budget baseline is set in this paper; therefore no size-performance ranking nor scaling-law proof is told.

BriLLM-MetaPred and the baseline have about 13.22M and 13.21M parameters respectively, both trained for 788 steps on 25,014,272 training tokens. Table 1 reports the training endpoints of two seeds, covering the full validation set; BriLLM-MetaPred uses the ADAPTIVE mode. The relative PPL drop is measured against the GPT baseline of the same seed.

Table 1: Model comparison on the BookCorpus validation set at equal training budget.

Seed	GPT baseline PPL ↓	BriLLM-MetaPred PPL ↓	Relative PPL drop
43	278.847	**229.451**	**17.71%**
44	279.355	**248.554**	**11.03%**

BriLLM-MetaPred lowers PPL by about 11%–18% across two seeds, showing that the continuous-execution mode secures competitive language-modeling performance at matched training budget.

With the training setup fixed and only the core mechanism changed, evaluation is made under the FROZEN mode, so that fast-parameter updates or state corrections at evaluation time do not veil the training difference. Table 2 reports the three-seed results.

Table 2: Core-mechanism ablation: three-seed mean and sample standard deviation; $\Delta\mathrm{PPL}$ is defined as the full model PPL minus the variant PPL.

Training variant	Validation PPL ↓	ΔPPL
Full model	36.700 ± 0.832	–
Modal nodes disconnected	37.887 ± 0.734	-1.187
Fast-parameter update and state correction off	36.657 ± 0.599	$+0.043$
Fixed-point objective removed	36.678 ± 0.906	$+0.022$

Disconnecting the modal nodes raises the validation PPL consistently, indicating that the modal node region M makes a measurable contribution to predictive performance; the limited magnitude of this increase shows that, with the modal nodes disconnected, the model still retains its overall predictive capability, which in turn indicates that the self-coordination node E can still independently support the model's global coordination function after losing M. Turning off the fast-parameter update and state correction, or lifting the fixed-point aim, has but a small effect upon the FROZEN-mode PPL, showing that the further curbs these give in training serve mainly the live phase of running, rather than static prediction under frozen inference.

Figure 3 gives the full 788-step training NLL curves of the two models at seed 42, the x-axis being cumulative training tokens. BriLLM-MetaPred's loss falls more smoothly, showing a steadier training dynamics.

The 129M model takes the sparse-auxiliary mode on THUCNews, run once at each of two training window sizes. Table 3 shows that both window sizes finish training and evaluation. Without equal GPT baseline, the present results take in no size-performance ranking nor scaling-law proof.

In sum, the results back the paper's aim on three fronts: (1) at matched training budget, BriLLM-MetaPred reaches PPL comparable to the equal-size GPT baseline across two seeds; (2) Disconnecting

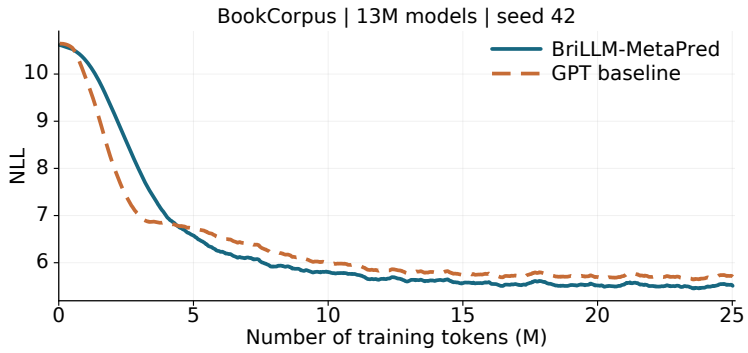

Figure 3: Training NLL of 13M BriLLM-MetaPred and GPT baseline on BookCorpus.

Table 3: Training endpoints of 129M BriLLM-MetaPred at two training window sizes, both at seed 42.

Input→output (tokens)	Training tokens	Validation tokens	BPC ↓	PPL ↓
$16 \rightarrow 16$	470,404,384	16,384	3.883050	**14.754158**
$512 \rightarrow 512$	133,826,560	7,635,968	4.766332	**27.215033**

the modal nodes consistently degrades prediction under FROZEN, indicating that the modal node region M makes a measurable contribution to predictive performance; the limited extent of this degradation shows that the model retains its overall predictive capability after the modal nodes are disconnected, which in turn suggests that the self-coordination node E can still independently support the model's global coordination function after losing M; (3) at about 129M parameters the model finishes training and evaluation and begets locally coherent text. These three results together back the three core traits of MetaPred—representation–prediction unity, endogenous self-supervision, and training–inference coupling—that the theory requires, and make up first proof of the engineering feasibility of the first language-model realization of MetaPred.

4 METAPRED AND THE FRONTIERS OF CURRENT AI RESEARCH

The derivation of the AGI mathematical theory (Anonymous, 2026) springs from a simple premise: the model must bear unbounded self-thought—thought about thought—letting infinite recursion and convergence. The result so drawn is MetaPred, a self-referential machine learning paradigm in the mathematical sense. Self-reference in computing must oft reckon with Gödel's incompleteness; yet Lambek's fixed-point lemma suffices to build a lawful, Gödel-consistent categorical self-referential frame (Lambek, 1968). That MetaPred asks the model realization to hold a fixed-point loss term is the necessary expression of this consistency; BriLLM-MetaPred heeds the ask.

This premise gives rise, upon our model, to three mechanistic conclusions. First, the fixed-point prediction aim, when external labels are gone, still gives a course of inner consistency, so that the model wins, of its own accord, the greatest possible reach in prediction. Second, the self-measurement mechanism simulates in real time within the dynamical-evolution context, lending to both training and inference a stronger give of efficiency and performance. Third, the single dynamical state makes training and inference walk together: both share one evolutionary skeleton, and no separate backward path for training or forward path for inference stands apart.

MetaPred is the first self-referential machine learning paradigm: one and the same dynamical state is at once the object of evolution and the carrier of that evolution. The model may lay upon its own state a straight endogenous shift of evolutionary course—marking the present course as one that cannot last, and starting a new course upon the same state. Self-referentiality, in engineering realization, shows itself on three planes: on the static-carrier plane, the self-coordination node E bears the model's representation of its own global state; on the dynamical plane, self-measurement takes the model's present state as input and weighs the model's own consistency at first hand; on the

loss plane, the fixed-point term asks the state evolution to reach a self-prediction steady state in one step. All three are realized by one and the same dynamics, so that the model finishes within itself the weighing and mending of its own evolutionary course. Creative thought thus wins its most straight mechanistic gate: a new course begins with the no-saying of the old, and no-saying and re-making alike are done upon the same state and by the same dynamics.

Let it be said plainly: the BriLLM-MetaPred shown here is no whole AGI system. By the AGI mathematical theory, a whole least AGI system holds at least four modules, each realized as a MetaPred model; among them is the super-modal module—language, natural language being the super modality that the theory foretells must be built in. The MetaPred frame, by the greatest self-symmetry of its machine learning elements, makes several MetaPreds, when nested, of their own accord form a MetaPred model that still bears the fixed-point loss aim, so showing its intrinsic self-similarity. What this paper realizes is but the super-modal module. The theory further asks that the super modality be built within the model, not set beside other modalities of perception and action at the interactive level; this hints that an AGI system must, in multimodal handling, place the language modality above other multimodal information. The theory also gives two mutually bound necessary premises: an AGI system must hold a language-model module; and an AGI system must realize a perception–action loop wholly self-consistent under unbounded self-reflection. Two complementary claims follow: an AGI system cannot be realized wholly outside a language model; and a language model alone cannot realize AGI. These two claims free the whole AGI system of hallucination by its own accord, for the perception–action loop at the interactive level gives the pure-symbolic thought a checkable outer curb and proof. The conclusion holds of the whole AGI system, not of the language-model module realized here.

Under this placing, MetaPred's mechanistic worth unfolds on two planes.

First, recursive self-improvement (RSI). MetaPred realizes representation and prediction through one unified dynamics, its dynamical state playing at once the role of architecture and of model state in the old sense. Upon this ground, the self-measurement mechanism is equal to a real-time self optimization, at the paradigmatically best level, of the model's own machine learning run at every timestep. If the AGI mathematical theory is right, this comes to realizing, at the intrinsic level of machine learning, the hope of RSI that current AI research chases.

Second, continual learning. MetaPred's symmetry shows itself in training–inference coupling: TRAIN, FROZEN, and ADAPTIVE are in essence three cases of the same class of dynamical evolution; even under the FROZEN mode, where no parameter is updated, the model still runs self-measurement in full. This is no added engineering ease, but a mechanism the theory asks of the model: every running mode holds the same evolutionary process as training, and so realizes, at the built-in-mechanism level, what the schools and the trade alike wish of continual learning in large models.

In the limit, the fixed-point prediction aim plays the last arbiter's part within the model's overall dynamics. When external labels fail, MetaPred's dynamics is still held by the ineliminable fixed-point aim; the model's thought thus reaches its state of last self-judgment. We make no claim that MetaPred has realized self-awareness or free will—yet this pure mathematical thought gives such mechanisms the mathematically and computationally best path.

Last, recall the derivation path of the natural-balance principle in the AGI mathematical theory (Anonymous, 2026). The natural-balance principle, using only the ratio functional of one energy (system cost) and one entropy (system information gain), likewise gives MetaPred as the best. Were that principle to hold, it would come to a straight claim, at the level of the mathematical optimization problem, that MetaPred is the most "thrifty" machine learning paradigm. MetaPred must take the general SiFu learning frame of loose coupling between carrier and dynamics—proving further that its model realization is not bound by scaling law: scaling law being the greatest mechanistic bottleneck of today's large GPT models. Model performance is loosed from model size, and the model takes, of its own accord, the best trade of spatiotemporal dynamical mechanisms from its architectural realization. Hence any model realization of MetaPred can loose input-output length from model size. Should this theoretical corollary hold, it brings a bright hope for the realization of a true AGI system: a runnable AGI system need not be a "large" model, nor need it take as its premise a memory holding all the world's encyclopedias. If this theoretical corollary holds, it carries a sharp corollary for the current scaling-centric path: general intelligence is not to be expected from scale alone, but from a model of a specific, mathematically mandated kind. BriLLM-MetaPred is a first instance of that kind.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing and presentation of this manuscript, including preparation of the ICLR LaTeX source; to retrieve and discover related work; and to assist in proving mathematical claims. The resulting text, citations, and mathematical arguments appear in the main paper and appendix. AI-generated output is not independent evidence for any claim. The authors take responsibility for the accuracy of the references, the validity of the proofs, and the final content of the paper.

REFERENCES

Anonymous. A mathematical theory of AGI. Submitted to ICLR 2027 for double-blind review, 2026.

Luc H Arnal and Anne-Lise Giraud. Cortical oscillations and sensory predictions. *Trends in Cognitive Sciences*, 16(7):390–398, 2012.

Andre M Bastos, W Martin Usrey, Rick A Adams, George R Mangun, Pascal Fries, and Karl J Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2): 127–138, 2010.

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. Coordination among neural modules through a shared global workspace. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9u1uF_B7X1M.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

Joachim Lambek. A fixpoint theorem in complete categories. *Mathematische Zeitschrift*, 103: 151–161, 1968.

Takahiro Noda, Tomoki Amemiya, Tomoyo I Shiramatsu, and Hirokazu Takahashi. Stimulus phase locking of cortical oscillations for rhythmic tone sequences in rats. *Frontiers in Neural Circuits*, 11:2, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael Kopp, et al. Hopfield networks is all you need. In *International Conference on Learning Representations (ICLR)*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Hai Zhao, Hongqiu Wu, Dongjie Yang, Anni Zou, and Jiale Hong. BriLLM: Brain-inspired large language model. *arXiv preprint arXiv:2503.11299*, 2025. Version 8.

A NOTATION AND GENERAL OPERATORS

All core symbols, operator actions, and basic rules are defined here once and for all.

A.1 Core Symbols and Indexing Rules

All tensor dimensions are marked at first sight, and indexing rules are the same throughout:

- **Nodes and neurons**: BriLLM-MetaPred is realized as a neural network, each node bearing a group of neurons with no free trainable link among the neurons within a node; the signal of a node is a d-dimensional real-valued vector, d being the neuron feature width of that node. The edges between nodes are directed, and two edges in opposite directions may coexist between a pair of nodes. Node signal flow, transform, and gating are all realized by standard neural operators—fully linked layers, layer normalization, GELU activation. Let the total number of nodes be $N = |V|$; the node signals form a tensor $\mathbf{S} \in \mathbb{R}^{N \times d}$.
- **Node split**: the node set V is split into 5 mutually exclusive subsets, and later operators by default act upon the split:

  I: the input node region, taking in external token projection vectors;
  O: the output node region, giving the signal source for prediction read-out;
  E: the self-coordination node, bearing global signal coordination;
  M: the modal node region, bearing learned modal reference vectors;
  W: the working node region, bearing dynamic computation signals;
  $U = M \cup W$: the adaptive internal node region;
  $C = \{E\} \cup M$: the coordination region, holding the self-coordination node and the modal nodes;
  $F = I \cup W \cup O$: the read-out auxiliary region, used for the pooled features of prediction read-out.
- **Graph-topology curb**: the model is defined upon a directed graph, V being the node set. The graph is **fully bidirected**: save for the curb that follows, edges run between any two nodes. The signal path between input region I and output region O passes through $M \cup W$ or E; this curb serves to bar a short circuit of the input signal into the read-out. In realization, no free edge matrix is kept for each pair of nodes; rather, talk between nodes is done by three node-axis mixing operators, acting upon the node subsets $I \cup W$, $I \cup C$, $W \cup C \cup O$ respectively; within each operator, two shared affine maps are used. Such a shape—"dense within a subset, in-order linked across subsets"—secures both the reach of the full link and a parameter count bound by the subset size rather than the square of the node-pair count.
- **Vocabulary**: $\mathcal{V}$ is the vocabulary, $|\mathcal{V}|$ the vocabulary size.
- **Evolutionary state**: $\mathbf{S}_{t,u} \in \mathbb{R}^{N \times d}$, where N is the total number of nodes and d the node dimension; t is the external token step, raised by 1 for every new token written; u is the internal executed evolutionary step, raised by 1 for every standard single-step evolution run.
- **Parameter-group index**: $q_u = u \bmod Q$, Q the total number of parameter groups; each parameter group bears a free evolutionary-transform weight, cyclically reused; a parameter group stores only transform parameters, and node signals are wholly borne by $\mathbf{S}_{t,u}$. Where no doubt may rise, q_u is cut to q below.
- **Three parameter families**: the learnable parameters are split by time scale into three families:

  Slow parameters $\boldsymbol{\theta}$: shared across sequences, batch-updated; holding the input projection matrix E_{in}, the output projection matrix $W_{\text{out}}, b_{\text{out}}$, the slot projection vectors $\{s_k, p_k\}$, the modal reference vectors $\{\mathbf{m}_k\}$, all weights of the five single-step-evolution stages, the read-out path weights, and the slow prediction parameters $p_{\boldsymbol{\theta}}$ and slow correction parameters $c_{\boldsymbol{\theta}}$.

  Fast parameters $\boldsymbol{\phi}_u$: sequence-local, updated by in-sequence events; holding the fast prediction parameters $p_{\boldsymbol{\phi}_u}$ and fast correction parameters $c_{\boldsymbol{\phi}_u}$; wholly reset at the sequence boundary. Slow prediction parameters $p_{\boldsymbol{\theta}}$ and fast prediction parameters $p_{\boldsymbol{\phi}_u}$ are together called the state predictors.

  Control parameters ψ: shared across sequences, updated by policy track, bound to the controller network.
- **Sequence runtime tuple**: $z_u = (\mathbf{S}_u, \boldsymbol{\phi}_u, \mathcal{H}_u, \mathcal{B}_u, q_u, \xi_u)$, updated with every executed evolutionary step u:

$\mathcal{H}_u$: causal historical observation buffer, storing state snapshots, residuals, errors, and other metadata of executed steps—weights are not stored;

$\mathcal{B}_u = (B_u^m, B_u^w)$: remaining budget, respectively for self-measurement and for fast-parameter update;

ξ_u: random-number-generator state, warranting the random streams of simulated evolution and true evolution stand wholly apart;

Where no doubt may rise, $\mathbf{S}_{t,u}$ is cut to $\mathbf{S}_u$. All runtime parts are reset once a sequence has been run (the sequence boundary); only $\boldsymbol{\theta}$ and the control parameters $\boldsymbol{\psi}$ are shared across sequences.

A.2 General Operators

The operations drawn into the model formulation are boxed as the following general operators:

- **Fully linked layer**: $\mathrm{Linear}_{W,b}(x) = Wx + b \in \mathbb{R}^m$, where $W \in \mathbb{R}^{m\times n}$ is the weight matrix, $b \in \mathbb{R}^m$ the bias vector.
- **Activation**: GELU activation $\mathrm{GELU}(\cdot)$ is used, acting on each part of the vector apart.
- **Softmax**: for $x \in \mathbb{R}^n$, the i-th part of $\mathrm{softmax}(x)$ reads $\exp(x_i)/\sum_j \exp(x_j)$.
- **Root mean square**: for a matrix X or a vector x, $\mathrm{RMS}(X) = \sqrt{\mathrm{mean}(X^2)}$.
- **Pooling operator**: $\mathrm{MeanPool}(G;\mathbf{S}) = \mathrm{mean}_G(\mathbf{S}) = \frac{1}{|G|}\sum_{n\in G}\mathbf{S}_n \in \mathbb{R}^d$, taking the mean over the node signals of a node subset G, used to draw global or local features.
- **Layer normalization**: $\mathrm{LN}(x) = \frac{x-\mathrm{mean}(x)}{\sqrt{\mathrm{var}(x)+10^{-6}}} \odot \gamma + \beta$, acting upon the channel dimension, with γ, β learnable parameters.
- **Gated mixing**:
$$\mathrm{GateMix}(\mathbf{Z}, w, F) = (1-w)\odot\mathbf{Z} + w \odot F(\mathbf{Z}), \tag{1}$$
where $w \in [0,1]$ is the part-wise gate weight, F the signal transform (by default $\mathrm{LN} \to \mathrm{Linear} \to \mathrm{GELU} \to \mathrm{Linear}$), the base unit of every MetaPred computation stage.
- **Bounding operator**:
$$\mathrm{Bounded}(\mathbf{v}, \mathbf{r}, c) = \mathbf{v} \cdot \frac{c\cdot\mathrm{RMS}(\mathbf{r})}{\mathrm{RMS}(\mathbf{v}) + c\cdot\mathrm{RMS}(\mathbf{r}) + 10^{-6}}, \tag{2}$$
scaling the size of $\mathbf{v}$ smoothly in step with the RMS of the reference $\mathbf{r}$, the scale factor falling as $\mathrm{RMS}(\mathbf{v})$ grows.
- **L2 normalization**: $\mathrm{norm}_2(x) = x/(\|x\|_2 + \epsilon)$, with ϵ a small constant.
- **Two clipping operations**:
  - Norm clipping:
$$\mathrm{ClipNorm}(g, c) = g \cdot \min\left(1, \frac{c}{\|g\|_2 + \epsilon}\right), \tag{3}$$
bounding the norm of a gradient or update to no more than c while holding its way, for slow-gradient clipping and fast-parameter update;
  - Scalar clipping:
$$\mathrm{ClipTo}(\rho, a, b) = \min(\max(\rho, a), b), \tag{4}$$
keeping a scalar within the span $[a, b]$, for clipping policy-probability ratios.
- **Normalized weighted mixing**:
$$\mathrm{WMix}(\{u_k\}, \{w_k\}) = \frac{\sum_k w_k u_k}{\sum_k w_k + 10^{-12}}, \tag{5}$$
summing a set of vectors by non-negative weights, normalized; used for modal-feature pooling.

- **Gated mapping**:
$$\mathrm{Gate}(x; W, b) = \sigma(Wx + b), \tag{6}$$
where σ is the part-wise sigmoid; used for coordination, broadcast, read-out, and kindred modules.
- **Relative MSE**:
$$\mathrm{RelMSE}(A, B) = \frac{\|A - \mathrm{sg}(B)\|_F^2 / n(A)}{\|\mathrm{sg}(B)\|_F^2 / n(A) + 10^{-6}}, \tag{7}$$
where $\mathrm{sg}(\cdot)$ is the stop-gradient operator, and $n(A)$ the number of scalar elements of A; this bars gradients from going back to the target end.
- **Measurement vector**:
$$\mathrm{MeasVec}(X) = [\mathrm{mean}(X), \mathrm{RMS}(X), \mathrm{mean}\,|X - \mathrm{mean}(X)|, \min(X), \max(X), \mathrm{mean}\,\mathbf{1}[X \text{ finite}]], \tag{8}$$
mapping any tensor to a 6-dimensional statistical feature, for trend supervision.
- **State summary**: for a state $\mathbf{S}$, define
$$s(\mathbf{S}) = \mathrm{first}_{d_h}[P_w(\mathrm{mean}_I(\mathbf{S}))\,; P_w(\mathrm{mean}_O(\mathbf{S}))\,; P_w(\mathbf{S}_E); P_w(\mathrm{mean}_U(\mathbf{S}))] \in \mathbb{R}^{d_h}, \tag{9}$$
where P_w is a channel projection, d_h the summary dimension, and first_{d_h} means taking the first d_h parts. The controller uses the control state summary $s_c(\mathbf{S}_u)$; the subscript c marks this summary as serving the controller input. It is a different function from the $s(\mathbf{S})$ used for trend computation.
- **Trend computation**: $\mathrm{Trend}(s, \Delta u, h) = \frac{s^{(h)}(\mathbf{S}') - s^{(h)}(\mathbf{S})}{\Delta u}$, where s is the state summary (as defined in Eq. (9)), $\mathbf{S}'$ the state after Δu steps of evolution from $\mathbf{S}$, $h = 1$ the first-order trend (speed) and $h = 2$ the second-order trend (haste), Δu being the evolutionary-step gap.

BriLLM-MetaPred holds two neural networks with free parameters. The state network is realized by a SiFu graph, bearing the slow parameters $\boldsymbol{\theta}$ and the fast parameters $\boldsymbol{\phi}_u$, and charged with the main state evolution; the controller network is a feedforward policy-value network, bearing the control parameters $\boldsymbol{\psi}$, and charged with setting the runtime conditions. The two interact through the state tensor and the runtime conditions.

B From the Primitive Prescribed by Mathematical Theory to the Engineered Architecture

What the AGI mathematical theory prescribes is a primitive, not a blueprint. It says the model must hold an endogenous fixed-point objective, that representation and prediction must issue from one and the same dynamics, that training and inference must accompany each other, and that a self-measurement mechanism must exist. It does not say the carrier must be a SiFu graph, that the network count must be two, or that the running modes must be of any particular number. Then which structural choice of BriLLM-MetaPred is forced by theory, which by present hardware, and which merely by the designer's taste? We now place every structural choice into one of three classes: rigid necessity from theory, sole viability under present constraints, and surface parameters irrelevant to paradigm identity. This classification is no bookkeeping. It yields a definite conclusion: *on existing silicon accelerators, BriLLM-MetaPred is the sole viable form that satisfies the four rigid necessities of the theory*.

Rigid necessity from theory. Four features of BriLLM-MetaPred are forced by the primitive itself. Any MetaPred realization must bear their kin, whatever carrier, scale, or hardware it selects.

First, a single evolving state tensor. If representation and prediction must issue from one and the same dynamics, no encoder-decoder split may stand between them; a single tensor $\mathbf{S}_{t,u}$ must bear both. Suppose representation were computed by one parameter block and prediction by another: the two would then issue from two dynamics, and representation-prediction unity would collapse on the spot. $\mathbf{S}_{t,u}$ is the direct writing of this necessity.

Second, an endogenous fixed-point loss. The theory demands a signal that measures the state's self-consistency under its own evolution, and no external label can supply it: an external label constrains

only the emitted distribution, not the state's trajectory. The loss $\mathcal{L}_{\text{fp}}$ and the self-measurement residual $\Delta_{\text{self}}^{(K)}$ are not design choices; they are the direct expression of the fixed-point requirement. Cancel them, and the model loses the self-supervision kernel that the theory demands.

Third, a self-measurement sub-process sharing the very same dynamical routine as the main process. The theory demands that the state measure itself. If the routine used for measurement differed from that used for evolution, the measurement would compare the state against a different dynamics, and the difference so obtained would no longer be the state's own inconsistency but a gap between two dynamics—of no avail to the theory's requirement. Hence Φ_q^{step} must serve both, distinguished only by the counter u. This is not to save parameters; it is to preserve the legitimacy of the measurement's meaning.

Fourth, training and inference must share one evolutionary skeleton. The theory's accompaniment requirement is this: the training path must exist equally at inference, and the inference path equally at training. Suppose independent training-forward and inference-forward paths existed: accompaniment would break, and self-measurement would of necessity degenerate into a training-only auxiliary loss, no longer carrying state correction at inference. It follows that any partition of the running process by the range of allowed parameter update or state correction must be a mode of the same dynamics. The theory does not mandate any particular number of modes; it mandates that each mode be an instance of the same skeleton. The count of three follows from two joint requirements—evaluation without update (FROZEN) and deployment with bounded update (ADAPTIVE)—each of which the theory's own premises require to be realizable.

The four above are the theory's four hard constraints. Every MetaPred realization must honor them; BriLLM-MetaPred honors them.

Sole viability under present constraints. Four further features are not forced by the primitive as such, but by the requirement that the primitive be executable on present hardware at tolerable cost. Given the present hardware and the four hard constraints of the theory, each admits but one viable form. On a different hardware substrate, the form may differ; yet what emerges on different hardware is another realization of the same MetaPred, not a refutation of the present one.

First, the SiFu carrier, not any hierarchical graph. The theory demands a carrier whose topological generality exceeds the specific architectures of current machine learning, while present accelerators execute only dense tensor operations efficiently. A hierarchical graph must maintain alignment between topology and tensor shape, which violates the theory's requirement of loose carrier-dynamics coupling. SiFu's fully bidirected, fully connected graph is the only structure satisfying both demands at once: greatest topological generality, and three node-axis mixing operators that map cleanly onto dense tensor operations. Other graph structures either lose generality or fail efficient execution. SiFu is thus no designer's preference, but the sole viable carrier under present hardware.

Second, two networks, neither one nor three. If the controller shared an encoder with the state network, the controller's policy gradient would flow straight back into the general representation; yet the theory's parameter-heterogeneity requirement rules that the control parameters' source of update (PPO) be fully isolated from those of the slow and fast parameters. Were a third network added, splitting the controller's policy head from its value head, the parameter-sharing principle of the actor-critic structure would be violated—policy and value must share one state representation, or advantage estimation ceases to be reliable. Two networks are therefore the sole decomposition: one bearing state dynamics, one bearing the policy-value map, with independent parameters and mutually sealed gradient paths.

Third, the five gated-mixing stages within a single-step evolution. The theory requires the state to be updated by one dynamics, but does not prescribe how the update be arranged. Any self-updating graph dynamics must accomplish four things: bring external information into the current state; let each node route information by its own role; compress global context into a coordination anchor; write the coordinated result back in localized form. The first demands input mixing; the second and third demand a coordination node and a broadcast; the write-back itself demands gated writing; and within the routing, each node must decide its leaning among the four functions—work, modal, fast, slow—which demands role assignment. The five stages are therefore not an interchangeable arrangement, but the minimal necessary factorization of these four functions upon the node graph.

The stages may be further subdivided, but they cannot be reduced: drop any one, and the functional demands can no longer be met at once.

Fourth, the bounded auxiliary pool and the measurement-tier system. The theory requires measurement and update to be finitely budgeted; otherwise the controller has no notion of cost, and a resource-allocation policy is unthinkable. This in turn demands a discrete set of tiers and a bounded pool from which the controller samples. The tier granularity must distinguish three functions: instant approximation (direct measurement), shallow simulation (short-range), and full parameter-group cycle (long-range). Skip is their lower bound. Four tiers are the minimal set that covers these three functions plus the zero signal; other schemes of comparable granularity are conceivable, but they must at least cover the same four functions—else either budget control or measurement depth is lost. Under the present realization, these four tiers are the sole viable set.

The four above are the sole viable forms once theory and present hardware are fixed. BriLLM-MetaPred adopts them not for convenience, but because constraints leave no alternative.

Surface parameters irrelevant to paradigm identity. What remains is compelled neither by theory nor by present hardware, and no matter what value it takes, it does not change whether the model is a MetaPred. Altering it changes no rigid necessity and no sole viable form; it is but a choice of scale, hyperparameter, or function class.

First, the concrete node count N and width d, and the partition sizes $|I|, |O|, |M|, |W|$. Any values that preserve the reachability required by the node-axis mixing operators and keep the parameter count tractable yield another instance of the same MetaPred.

Second, the parameter-group count Q, the temperature τ, the fourfold role split, the number and values of correction strengths, and the loss weights $\lambda_{\text{int}}, \lambda_p, \lambda_f, \lambda_d, \lambda_c, \lambda_s$. These are all hyperparameters: they touch neither the four hard constraints of the theory nor the four sole viable forms above. Their present settings reflect tuning, not derivation.

Third, the concrete neural operators—GELU, layer normalization, the specific two-layer map of the state predictors—and the initialization scheme. The theory constrains only the functions these operators must perform (e.g., GELU must be a smooth nonlinearity, layer normalization must remove channel mean and scale), not which function class realizes them. Swap in other operators performing the same functions, and the model remains a MetaPred.

The worth of this threefold partition lies not in bookkeeping. It tells us three definite facts. First, where the theory tightens: four rigid necessities, which no MetaPred realization may violate. Second, where present hardware tightens: four sole viable forms, which on existing accelerators cannot be replaced. Third, where the designer is free: three kinds of surface parameters, whose values affect performance only, not paradigm identity.

From this a conclusion far weightier than any ledger follows. *The paradigm identity of BriLLM-MetaPred is fully determined by the four rigid necessities and the four sole viable forms, and is independent of every surface parameter.* A future MetaPred realization on different hardware, or with different tuning, may depart from every surface parameter, may even depart from every sole viable engineering form above, and yet remain a MetaPred—provided it honors the four rigid necessities. It will then automatically satisfy the representation-prediction unity, the endogenous self-supervision, the training-inference accompaniment, and the self-referential fixed point that the theory demands. This is precisely the shape of the bridge between the AGI mathematical theory and any concrete machine: the primitive is one, the realizations may be many, and the present BriLLM-MetaPred is the *sole* viable realization on existing silicon accelerators that satisfies every hard constraint of the theory.

This conclusion yields, at the same time, a definite judgment on the present scaling-centric path. Should the above corollary hold, general intelligence cannot emerge from scale: raising scale only amplifies the values of surface parameters, touching neither the four rigid necessities nor the four sole viable forms. Were scaling truly to bring general intelligence, it would be a mere accident of engineering—at some point it would have chanced upon a mechanism satisfying the four rigid necessities, and the occurrence of such an accident is guaranteed by no scaling law. What can truly support general intelligence is only a specific model mandated by mathematics; BriLLM-MetaPred is the first instance of that kind.

C THE TWO ELEMENTARY DYNAMICAL PROCESSES: FULL DEFINITION

MetaPred's dynamical build holds but two elementary processes: **main state evolution** and **self-measurement**. Every other part of the model is but mending or reading of these two. Both share one standard dynamical routine Φ_q^{step}: main state evolution is that routine run upon the runtime state, moving the evolutionary step u; self-measurement, under the controller's sway, recurses that routine upon a copy of the same state, leaving the evolutionary step u as it was. Thus self-measurement makes up a second-order recursive dynamical sub-process kept by the controller. Main state evolution, by the way external input is fed in, takes two modes: **input-bearing evolution** and **input-free evolution**. Input-bearing evolution, by the content of external input, further splits into **encoding evolution** and **generation evolution**; input-free evolution is **autonomous evolution**. Prediction read-out draws straight from the present evolutionary state, moving not u, and is no part of the evolutionary process.

Upon these two elementary processes, the whole run of BriLLM-MetaPred may be split into the six layers that follow:

- **External interface layer**: the setting-off condition for main state evolution. It takes in input tokens, writes their projection vectors into the input node region I, and sets off encoding evolution;
- **Core dynamics layer**: main state evolution itself. It updates the state upon the single-step evolution Φ_q^{step}, backing autonomous and generation evolution;
- **Self-measurement layer**: self-measurement itself. It builds, from one and the same state, two simulations of unlike step counts—or gives an approximate residual by direct measurement—and takes the difference of the two as the endogenous supervision signal;
- **Decision layer**: the scheduling mending of the two elementary processes. The controller, from the causal state summary and the historical observations, picks the present step's measurement tier, correction strength, and fast-parameter update policy;
- **Read-out layer**: the read-out mending of main state evolution. It draws features from the output region O and the coordination region C of the evolutionary state, lays on a semantic residual, a structural residual, and slot alignment, and begets the vocabulary distribution through the output projection;
- **Update layer**: the learning mending of self-measurement. It updates the slow parameters $\boldsymbol{\theta}$, fast parameters $\boldsymbol{\phi}_u$, and control parameters $\boldsymbol{\psi}$ apart, the gradient paths wholly apart.

Among the six layers, the external interface, core dynamics, self-measurement, read-out, and update layers all act straight upon one and the same state tensor, sharing one and the same evolutionary path. The decision layer is realized by a free controller network: the controller reads a causal state summary from the state tensor and auxiliary metadata, and gives out actions that shape the conditions of later evolution.

MetaPred's way of running is not free to pick; the three running modes are the straight result of the single dynamical state and the heterogeneity of parameter time scales. All three modes share one evolutionary skeleton, differing only in the span of allowed operations and parameter updates. The differences among the three modes appear in Table 4; therein, evaluation metrics are computable under all three modes, yet are not used for parameter update under FROZEN and ADAPTIVE.

Table 4: Comparison of the three running modes

Feature	TRAIN	FROZEN	ADAPTIVE
External task loss for parameter update	✓	✗	✗
Self-measurement diagnostic	✓	✓	✓
State correction	✓	✗	✓
Fast-parameter update	✓	✗	✓
Slow-parameter update	✓(by batch)	✗	✗
Control-parameter update	✓(delayed PPO)	✗	✗
Sequence runtime reset	sequence boundary	sequence boundary	sequence boundary

The three modes have unlike roles:

TRAIN mode serves training: external task loss, self-measurement, fast-parameter update, state correction, slow-parameter update, and control-parameter update are all on.

FROZEN mode serves evaluation and self-diagnosis: all parameters and state corrections are frozen, external task loss may be computed on the validation set to watch the model's capability, while self-measurement is run in full and the residual signal is given out; since self-measurement needs only state evolution itself, the residual under this mode is a straight measure of the model's inner coherence.

ADAPTIVE mode serves the mock of real deployment: it leans on internal signals only, lets sequence-local fast-parameter update and state correction, and keeps the slow parameters $\boldsymbol{\theta}$ and control parameters $\boldsymbol{\psi}$ fixed. All three modes run dynamics and self-measurement in full, differing only in the span of allowed update and correction. All sequence runtime states are reset at the sequence boundary; only $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ are shared across sequences.

The split into three modes follows straight from two theoretical curbs of MetaPred, not from the freedom of engineering optimization: first, the single state needs training and inference to share one evolutionary skeleton, so that the three modes must take one and the same dynamics; second, endogenous supervision and training–inference coupling need self-measurement to run in full under every mode, and hence the differences among the three modes are kept to the span of enabled parameter update and state correction. Where a later section touches parameter update or state correction, it shall say which running mode the operation falls to. The next two sections define the two elementary processes in turn: main state evolution and self-measurement.

C.1 MAIN STATE EVOLUTION

All machine learning operations of the state network rest upon the evolutionary process over the main state space, a standard signal-dynamics routine.

As to **single-step evolution**: $\Phi_q^{\text{step}}(\cdot;\chi) : \mathbb{R}^{N\times d} \to \mathbb{R}^{N\times d}$, it holds five gated-mixing stages run in order. Here χ is the historical-dependency condition set for the evolutionary step, holding the parameter-group index, the invocation stage, the content of the input window, and the remaining budget, among other runtime conditions. All five stages share one pattern—gated mixing (as defined in Eq. (1)). The mathematical definition of every stage now follows.

1. **Input mixing.** Apply the node-axis mixing operator to three node subsets in turn,
$$\mathcal{M}_G^q(Z) = Z + B_G^q(\mathrm{GELU}(A_G^q(\mathrm{LN}(Z)))), \tag{10}$$
giving $Z^3 = \mathcal{M}^q_{W\cup C\cup O}(\mathcal{M}^q_{I\cup C}(\mathcal{M}^q_{I\cup W}(\mathbf{S} + e_{\text{stage}})))$. The node subsets upon which the three node-axis mixing operators act hold no straight link between I and O; hence the signal path from input region to output region passes through $M \cup W$ or E. A_G^q, B_G^q are two fully linked layers on the node axis.
2. **Role assignment.** For each internal node $n \in U$, compute a four-dimensional role weight
$$\pi_n = \mathrm{softmax}\left(\frac{a_n^{\text{role}} + \Delta_n}{\tau}\right), \qquad \Delta_n = \alpha\sigma(\gamma)c([Z_n^3, Z_E^3]) + b_{n,q}.$$
Here $a_n^{\text{role}} \in \mathbb{R}^4$ is a per-node, per-parameter-group learnable role prior, $\alpha, \gamma \in \mathbb{R}$ learnable scalars, σ the sigmoid, c a shared two-layer map $\mathrm{Linear}(2d, 2d) \to \mathrm{GELU} \to \mathrm{Linear}(2d, 4)$, and τ the temperature constant, taken as $\tau = 1$. $b_{n,q} \in \mathbb{R}^4$ is a per-node, per-parameter-group learnable bias; every node bears one $\mathbb{R}^4$ bias per parameter group.
The four parts stand in order for work, modal, fast, and slow. Work and modal parts mix two shared channel transforms:
$$\bar{Z}_n = Z_n^3 + \frac{\pi_n^{\mathrm{w}}}{\pi_n^{\mathrm{w}} + \pi_n^{\mathrm{m}}} F_{\mathrm{w}}^q(Z_n^3) + \frac{\pi_n^{\mathrm{m}}}{\pi_n^{\mathrm{w}} + \pi_n^{\mathrm{m}}} F_{\mathrm{m}}^q(Z_n^3),$$
where $F_{\mathrm{w}}^q, F_{\mathrm{m}}^q$ are two shared channel transforms of the shape $\mathrm{LN} \to \mathrm{Linear} \to \mathrm{GELU} \to \mathrm{Linear}$.

3. **Coordination aggregation.** The self-coordination node E squeezes global information into Z_E^c:

$$\mathbf{m}_k^* = (1-\alpha_m)\mathbf{m}_k + \alpha_m \,\mathrm{norm}_2\left(\sum_i \omega_{k,i}\bar{Z}_i\right), \qquad \omega_{k,i} = \mathrm{softmax}_i(\mathbf{m}_k^\top \bar{Z}_i),$$

$$Z_E^I = \bar{Z}_E + g_E(\bar{Z}_E W_E - \bar{Z}_E), \qquad c_E = \sum_k \mathrm{softmax}_k\big((Z_E^I)^\top \mathbf{m}_k^*\big)\mathbf{m}_k^*,$$

$$Z_E^c = \bar{Z}_E + \mathrm{bounded}\big((Z_E^I - \bar{Z}_E) + g_c c_E,\, Z_E^I,\, c_E^{\max}\big).$$

Here $\{\mathbf{m}_k\}_{k=1}^{|M|}$ are the learnable modal reference vectors among the slow parameters; $\mathbf{m}_k^*$ is the dynamic reference used for the present coordination, the slow-parameter $\mathbf{m}_k$ holding its value; α_m, g_E, g_c are learnable scalars, and $c_E^{\max}$ an upper-bound constant.

4. **Broadcast distribution.** The self-coordination node sends a broadcast offset to each $n \in U$:

$$m_U = \mathrm{WMix}\big(\{\bar{Z}_i\}_{i\in U}, \{\pi_i^{\mathrm{m}}\}_{i\in U}\big), \qquad b_n^{\mathrm{br}} = \tanh\big(W_b[\mathrm{LN}([Z_E^c, m_U]); \mathrm{LN}(\bar{Z}_n)]\big),$$

$$\delta_n^b = \mathrm{bounded}\big(\kappa_b \sigma(\gamma_b) c_b \pi_n^{\mathrm{w}} b_n^{\mathrm{br}},\, b_n^{\mathrm{br}},\, c_b^{\max}\big), \qquad Z_n^b = \bar{Z}_n + \delta_n^b.$$

Here $\kappa_b, c_b, c_b^{\max}$ are the strength, control, and upper-bound constants of the broadcast. The broadcast target is the nodes in U; I, O, E hold their values.

5. **Gated writing.** Each node writes the middle signal back to the state by fast and slow gates:

$$h_n = \mathrm{Gate}\Big([\mathbf{S}_n, \widehat{Z}_n]; G_{\mathrm{f}}\Big)\,\rho_n^{\mathrm{f}} + \mathrm{Gate}\Big([\mathbf{S}_n, \widehat{Z}_n]; G_{\mathrm{s}}\Big)\,\rho_n^{\mathrm{s}}, \qquad \mathbf{S}'_n = (1-h_n)\mathbf{S}_n + h_n \widehat{Z}_n,$$

Here $G_{\mathrm{f}}, G_{\mathrm{s}}$ are two fully linked layers, $\rho_n^{\mathrm{f}} = \pi_n^{\mathrm{f}}$ and $\rho_n^{\mathrm{s}} = \pi_n^{\mathrm{s}}$ are taken respectively from the fast and slow parts of the four-dimensional role weight at the role-assignment stage, and $\widehat{Z}_n$ is the node middle signal after the first four stages. The output $\mathbf{S}' = \Phi_q^{\text{step}}(\mathbf{S}; \chi)$ is the result of one-step evolution.

Single-step evolution updates only the node signal $\mathbf{S}$, holding the parameter-group index q, the slow parameters $\boldsymbol{\theta}$, and the historical buffer $\mathcal{H}_u$ as they were. Every evolutionary step feeds a stage projection vector e_{stage} as the invocation stage (encoding, autonomous, generation), to mark the present stage of evolution. $e_{\text{stage}} \in \mathbb{R}^d$ is one of three learnable vectors, answering respectively to the three invocation stages of encoding, autonomous, and generation; every node adds the same e_{stage} during input mixing. State read-out is an operation free of the evolutionary process: it draws the prediction distribution straight from the present evolutionary state, moving not the evolutionary step u.

The single-step evolution above is the concrete instantiation of the MetaPred paradigm upon the node graph, its shape fixed by the five-stage signal lifecycle. The five stages answer strictly to the whole lifecycle of the signal—from input, through inner coordination, to output: input mixing wins that external token information enters the current state; role assignment lets each node of its own will lean among the four functions "work / modal / fast / slow," realizing a lithe reshaping of representation; coordination aggregation squeezes global context through the self-coordination node, giving the whole state one steady "attention" anchor; broadcast distribution sends the coordination signal back to the inner nodes, making global information local; gated writing rules the size of the state update by fast and slow paths, winning the steadiness and amendability of evolution. All five stages rest upon one and the same gated-mixing operator, bearing out the chief tenet of MetaPred, "one single, ever-evolving dynamical state"—casting aside model designs with free encoder or decoder, every operation being an iterative transform upon the same state tensor, along one and the same evolutionary path.

Continuous K-step evolution is defined as the composite of parameter groups, called **multi-step evolution**:

$$\Phi_q^{(K)} = \Phi_{q+K-1}^{\text{step}} \circ \Phi_{q+K-2}^{\text{step}} \circ \cdots \circ \Phi_q^{\text{step}}$$

When $K = Q+1$, $\Phi_q^{(Q+1)}$ runs from phase q for $Q+1$ steps to reach phase $q+1$; in like wise, the Q-step composite from phase $q+1$, $\Phi_{q+1}^{(Q)} = \Phi_{q+Q} \circ \cdots \circ \Phi_{q+1}$, makes up a whole parameter-group cycle, denoted $\mathcal{C}_{q+1} = \Phi_{q+1}^{(Q)}$, answering to the simulated measurement with $K = Q+1$; multi-step

evolution leaves the true evolutionary-step counter u as it was, and serves only simulation and self-measurement.

All true evolution (encoding, autonomous, generation) moves u; all simulated evolution (self-measurement, preview, alternative weighing) leaves u as it was, and thereby the runtime state and the simulated state stand wholly apart. Self-measurement is of this latter kind: it recurses upon one and the same standard dynamical routine Φ_q^{step}, its invocation timing, simulation depth, and measurement tier all picked by the controller.

C.2 The Self-Measurement Process

This section defines MetaPred's second elementary process: self-measurement, the second-order recursive dynamical sub-process kept by the controller. It shares one and the same standard dynamical routine Φ_q^{step} with main state evolution; the difference is that main state evolution runs this routine upon the runtime state and moves the evolutionary step u, whereas self-measurement, under the controller's sway, recurses upon a copy of the same state and leaves u as it was. Self-measurement is therefore nested above main state evolution, not a static loss term standing apart from the dynamics.

The base thought of self-measurement is this: at an executed evolutionary step, from one and the same state, build two simulations of unlike step counts, and take the difference of their endpoints as the signal. The one-step simulation is the model's straight prediction of the next step; the K-step simulation is the piled-up result of the dynamics over several evolutions; their difference measures the discord of the dynamics at finite depth. This signal has four uses: to train the shared dynamics, to supervise the direct measurement, to drive state correction, and to give an intrinsic-coherence diagnostic. The first three answer respectively to parameter update under TRAIN or ADAPTIVE; the fourth leans on no parameter update, and serves as the model's self-diagnostic reading under FROZEN.

The controller may pick among four measurement tiers: skip, direct measurement, short-range simulation, and long-range simulation (the latter two being splits of simulated measurement by depth). Skip answers to zero signal out; direct measurement gives an approximate residual from the direct-measurement parameters alone; simulated measurement really runs state simulation, its depth set by the parameter K. The direct-measurement parameters hold the slow prediction parameters $p_{\boldsymbol{\theta}}$ and slow correction parameters $c_{\boldsymbol{\theta}}$ among the slow parameters, together with the fast prediction parameters $p_{\boldsymbol{\phi}_u}$ and fast correction parameters $c_{\boldsymbol{\phi}_u}$ among the fast parameters. When $K = 2$ (marked short-range simulation), the simulation supervises the direct measurement; when $K = Q + 1$ (marked long-range simulation), the simulation answers to one whole parameter-group cycle and serves the deep diagnostic. Short-range and long-range simulations use one and the same mechanism, differing only in the value of K.

Since state-evolution simulation asks extra computation, the model also learns the direct-measurement parameters, with the controller giving out measurement and update budgets. This section follows the run order of one inner-evolution call to show how these mechanisms work in tune. At call, the controller picks the present measurement tier and update ask from the current state and historical observations; then gets the residual as the chosen tier; if a fast-parameter update is asked, the direct measurement must be wrought afresh with the updated parameters; the model saves the prediction of the next state and runs one true evolution; last it lays on state correction as asked and updates to the next state; once the true result comes in, the model notes the prediction error and the measurement information, giving feedback for later calls.

Simulated measurement Let χ_u be the condition instance of evolutionary step u, and χ_u^{meas}, χ_u^{real} the conditions used for simulated measurement and for true evolution respectively. At the executed evolutionary step u, freeze χ_u^{meas} and build two certain simulations:

$$\mathbf{S}_{\text{pred}} = \Phi_q^{\text{step}}(\mathbf{S}_u; \chi_u^{\text{meas}}) \qquad \text{(1-step simulation)}$$

$$\mathbf{S}_{\text{sim}}^{(K)} = \Phi_q^{(K)}(\mathbf{S}_u; \chi_u^{\text{meas}}) \qquad \text{(K-step simulation)}$$

Their difference is the self-measurement residual:

$$\Delta_{\text{self}}^{(K)}(u) = \mathbf{S}_{\text{sim}}^{(K)} - \mathbf{S}_{\text{pred}}.$$

Here Φ_q^{step} and $\Phi_q^{(K)}$ are one and the same standard dynamical routine; self-measurement brings in no new dynamical operator, only recursing upon the same dynamics.

The residual bears a term-by-term split

$$\Delta_{\text{self}}^{(K)}(u) = \sum_{k=1}^{K-1} \left(\mathbf{S}_{\text{sim}}^{(k+1)} - \mathbf{S}_{\text{sim}}^{(k)} \right), \qquad \|\Delta_{\text{self}}^{(K)}(u)\|_F \le \sum_{k=1}^{K-1} \|\mathbf{S}_{\text{sim}}^{(k+1)} - \mathbf{S}_{\text{sim}}^{(k)}\|_F,$$

where $\mathbf{S}_{\text{sim}}^{(1)} = \mathbf{S}_{\text{pred}}$.

When $K = Q + 1$, the K-step simulation answers to one whole parameter-group cycle starting from $\mathbf{S}_{\text{pred}}$. Defining the Q-step composite from phase $q+1$ as $\mathcal{C}_{q+1} = \Phi_{q+1}^{(Q)}$, one has

$$\Delta_{\text{self}}^{(Q+1)}(u) = \mathcal{C}_{q+1}(\mathbf{S}_{\text{pred}}) - \mathbf{S}_{\text{pred}} = \Phi_q^{(Q+1)}(\mathbf{S}_u) - \mathbf{S}_{\text{pred}}.$$

The residual meets the telescoping-split property.

The self-measurement loss uses a sequence-level normalization:

$$d_{\text{self},u} = \frac{\|\Delta_{\text{self}}^{(K)}(u)\|_F}{Nd}, \qquad \mathcal{L}_{\text{self}} = \frac{1}{|\mathcal{D}_{\text{self}}|} \sum_{u \in \mathcal{D}_{\text{self}}} d_{\text{self},u}^2. \tag{11}$$

Here $\mathcal{D}_{\text{self}}$ is the set of executed simulated-measurement steps; the $K = Q + 1$ residual takes no part in direct-measurement parameter calibration.

State correction from self-measurement If the controller asks for state correction, upon the true successor $\mathbf{S}_{u+1}^{\text{pre}} = \Phi_q^{\text{step}}(\mathbf{S}_u; \chi_u^{\text{real}})$, lay on a correction term:

$$\delta\mathbf{S}_u = \eta_{\text{eff}} \widetilde{\Delta}_{\text{self}}^{(K)}(u), \qquad \mathbf{S}_{u+1} = \mathbf{S}_{u+1}^{\text{pre}} + \delta\mathbf{S}_u. \tag{12}$$

Here $\widetilde{\Delta}_{\text{self}}^{(K)}(u)$ is the available simulated or direct-measurement residual, and η_{eff} is the effective value of the controller's asked strength after scaling by the amplitude caps. Let the non-negative correction strength answering to controller action a_u^{corr} be η_u^{req}, the state change of true evolution be $\mathbf{S}_{u+1}^{\text{pre}} - \mathbf{S}_u$, and the two amplitude caps be

c_{abs}: the absolute cap on the whole size of the correction;

c_{rel}: the cap on the correction size relative to the present true-evolution size.

Measuring the whole size of a tensor by $\mathrm{RMS}(\cdot)$, and taking $\epsilon > 0$ as a small constant to bar a zero denominator, the effective strength is

$$\eta_{\text{eff}} = \min \left\{ \eta_u^{\text{req}}, \frac{c_{\text{abs}}}{\mathrm{RMS}(\widetilde{\Delta}_{\text{self}}^{(K)}(u)) + \epsilon}, \frac{c_{\text{rel}}\ \mathrm{RMS}(\mathbf{S}_{u+1}^{\text{pre}} - \mathbf{S}_u)}{\mathrm{RMS}(\widetilde{\Delta}_{\text{self}}^{(K)}(u)) + \epsilon} \right\}. \tag{13}$$

The three bounds answer respectively to the controller's ask, the absolute-size limit, and the relative-size limit; the least makes the actual correction. With no correction ask, or where the running mode bars correction, or where the present residual is not to hand, the true-evolution result is used:

$$\mathbf{S}_{u+1} = \mathbf{S}_{u+1}^{\text{pre}}. \tag{14}$$

The corrected state lies upon the segment between the two simulation endpoints (true only for simulated residuals).

Direct-measurement parameter supervision Direct measurement runs two prediction stages atop the direct-measurement parameters.

First, predict the successor before correction. Write the current state as $\mathbf{S}_u$ and the runtime condition as χ_u; then

$$\widehat{\mathbf{S}}_{u+1}^{\text{pre}} = \mathbf{S}_u + p_{\boldsymbol{\theta}}(\mathbf{S}_u, \chi_u) + p_{\boldsymbol{\phi}_u}(\mathbf{S}_u).$$

Here $p_{\boldsymbol{\theta}}$ is shared across sequences, while $p_{\boldsymbol{\phi}_u}$ is sequence-local. In the actual implementation, the slow prediction map is composed of a base prediction map and a slow prediction adapter:

$$p_{\boldsymbol{\theta}}(\mathbf{S}_u, \chi_u) = p_{\boldsymbol{\theta}}^{\text{base}}(\mathbf{S}_u, \chi_u) + p_{\boldsymbol{\theta}}^{\text{adapt}}(\mathbf{S}_u).$$

The base prediction map $p_{\boldsymbol{\theta}}^{\text{base}}$ adopts $\text{LN} \to \text{Linear}(d, d) \to \text{GELU} \to \text{Linear}(d, d)$; its input combines the stage, parameter group, and prediction step-size information from the runtime conditions, and its output, after scaling, forms the base prediction increment. The parameters of this map belong to the trainable slow parameters $\boldsymbol{\theta}$.

The slow prediction adapter $p_{\boldsymbol{\theta}}^{\text{adapt}}$ and the fast prediction map $p_{\boldsymbol{\phi}_u}$ both adopt a bottleneck structure

$$\text{LN} \to \text{Linear}(d, r) \to \text{GELU} \to \text{Linear}(r, d),$$

where r is the bottleneck dimension of the adapter. The above maps act node-wise and share parameters across nodes; these maps themselves do not additionally perform neighborhood aggregation along the node axis. The slow prediction adapter uses slow parameters shared across sequences; the fast prediction map uses fast parameters of the current sequence. Therefore, the complete state prediction is obtained by summing the current state, the base prediction increment, the slow prediction adaptation increment, and the fast prediction adaptation increment.

Second, estimate the residual from the predicted state got in the first stage:

$$\widehat{\Delta}_{\text{self},u+1}^{(2)} = c_{\boldsymbol{\theta}}(\widehat{\mathbf{S}}_{u+1}^{\text{pre}}) + c_{\boldsymbol{\phi}_u}(\widehat{\mathbf{S}}_{u+1}^{\text{pre}}).$$

Here $c_{\boldsymbol{\theta}}$ and $c_{\boldsymbol{\phi}_u}$ both take the predicted state $\widehat{\mathbf{S}}_{u+1}^{\text{pre}}$ as input, respectively giving a slow-parameter correction and a fast-parameter correction, whose sum forms the residual estimate of the direct measurement. Both maps act node-wise and share parameters across nodes, adopting

$$\text{LN} \to \text{Linear}(d, r) \to \text{GELU} \to \text{Linear}(r, d),$$

and outputting a d-dimensional residual estimate at each node. Here $c_{\boldsymbol{\theta}}$ uses shared slow parameters, and $c_{\boldsymbol{\phi}_u}$ uses sequence-local fast parameters. The residual prediction process does not directly overwrite the true successor state $\mathbf{S}_{u+1}^{\text{pre}}$; after passing the direct-measurement availability check, its output is used for state correction under the aforementioned strength and amplitude constraints.

Let the one-step simulation of observation i be $\mathbf{S}_{\text{pred},i}$ and its $K = 2$ residual $\Delta_{\text{self},i}^{(2)}$. Let $\mathcal{D}_{K=2}$ be the set of executed $K = 2$ simulated-measurement steps. From the definition of $\mathcal{D}_{\text{self}}$, $\mathcal{D}_{K=2} \subseteq \mathcal{D}_{\text{self}}$. The direct-measurement parameter supervision loss is:

$$\mathcal{L}_{\text{direct-self}} = \mathbb{E}_{i \in \mathcal{D}_{K=2}} \frac{1}{2} \left[\text{RelMSE}(\widehat{\mathbf{S}}_i^{\text{pre}}, \mathbf{S}_{\text{pred},i}) + \text{RelMSE}(\widehat{\Delta}_{\text{self},i}^{(2)}, \Delta_{\text{self},i}^{(2)}) \right] \tag{15}$$

using only $K = 2$ simulated measurements that have come in, as supervision; the $K = Q+1$ residual takes no part in direct-measurement parameter training.

The availability of a direct measurement is fixed by the check made upon previously noted simulated measurements. The check covers: whether invocation stage, parameter-group index, fast-parameter version, and simulation depth square with the present ask; whether the note still lies within the timeliness window; and whether the estimation error falls below the relative threshold. Should any condition fail, that direct measurement is marked not to hand, and the present correction falls back upon a simulated residual or is skipped. The $K = 2$ direct measurement and the $K = Q + 1$ simulated result fall to unlike measurement tiers; the $K = Q + 1$ residual is got by the answering simulated measurement.

If, in the present call, the controller asks for a fast-parameter $\boldsymbol{\phi}_u$ update, that update shifts the outputs of $p_{\boldsymbol{\phi}_u}$ and $c_{\boldsymbol{\phi}_u}$. Therefore, whenever a direct measurement is picked, the first and second prediction stages must be run afresh after the $\boldsymbol{\phi}_u$ update, and the check above done once more. The simulated-measurement residual stands free of fast-parameter update, holding its value.

Let it be stressed: the size of the direct-measurement parameters' output and the size of the true residual stand free of one another. Direct measurement is but an approximation of the present residual, its trust won by the check made from simulated measurements. Whatever the present call, whether or not it lays on state correction, simulated measurement takes part in the intrinsic-coherence learning

and gives supervision for later direct measurement; hence simulated measurement is not to be spared in the whole training flow.

$\widehat{\mathbf{S}}_{u+1}^{\text{pre}}$ and $\mathbf{S}_{u+1}^{\text{pre}}$ may differ: direct measurement predicts under the runtime condition before the action, while true evolution makes the state under the runtime condition after the action.

The self-measurement mechanism runs within a conditional-convergence frame. Only when the whole parameter-group cycle $\mathcal{C}_{q+1} = \Phi_{q+1}^{(Q)}$ (leaning on the slow parameters $\boldsymbol{\theta}$ and the runtime condition χ) is a contraction upon the closed subset $\mathcal{X}$,

$$\|\mathcal{C}_{q+1}(A) - \mathcal{C}_{q+1}(B)\|_F \le L\|A - B\|_F, \quad L < 1,$$

does the Banach fixed-point theorem warrant a sole fixed point $\mathbf{S}^\star$, and

$$\|\mathcal{C}_{q+1}^m(\mathbf{S}) - \mathbf{S}^\star\|_F \le L^m\|\mathbf{S} - \mathbf{S}^\star\|_F.$$

This conclusion is conditional; the convergence of a trained model is fixed by its concrete parameters and computational dynamics.

D Working Architecture: Detailed Specification

This section describes the external interface layer, the read-out layer, and the decision layer among MetaPred's six-layer structure. These three layers are all correction or reading of the two elementary processes: the external interface layer is the triggering condition of main state evolution; the read-out layer is the read-out correction of main state evolution; the decision layer is the scheduling correction of the two elementary processes.

D.1 External Interface Layer: Token Input and State Writing

External tokens are written into the evolutionary state in three steps:

1. **Projection lookup**: for the input token sequence $x_{1:T}$ ($x_t \in \mathcal{V}$), obtain the token projection vector $e_t = E_{\text{in}}[x_t]$ through the shared input projection matrix $E_{\text{in}} \in \mathbb{R}^{|\mathcal{V}|\times d}$. E_{in} is randomly initialized by a Xavier uniform distribution; the input projection matrix is a subset of the slow parameters $\boldsymbol{\theta}$.
2. **Sliding-window update**: the input region I maintains an ordered window of capacity $|I|$. When a new token projection vector arrives, the oldest projection vector is discarded from the left end and the new one appended to the right:

   $$H_t = (e_{t-|I|+1}, \ldots, e_{t-1}, e_t)$$

   When $t < |I|$, the left end is padded with zero vectors; token information older than the window capacity is compressed by dynamical evolution into the working region W and the modal region M, and kept as long-term memory.
3. **State writing**: the window content covers the input-region state by slot, and two learnable slot projection vectors s_k and p_k are superposed:

   $$(\mathbf{S}_t^{\text{write}})_{i_k} = e_{t-|I|+k} + s_k + p_k, \quad k = 1, \ldots, |I|$$

   Here i_k is the index of the k-th node in the input region; s_k and p_k correspond respectively to the slot embedding table and the position embedding table in the realization. Both are learnable parameters, initialized by a normal distribution $\mathcal{N}(0, 0.02^2)$.

Evolution triggering rule: state writing advances only the token step t, leaving the evolutionary step u unchanged; once writing is done, K_{in} steps of encoding evolution Φ_q^{step} are triggered; once the context is wholly written, K_{ev} steps of autonomous evolution are triggered (with no new token injected); in the generation stage, one step of generation evolution is triggered for every target token read. Among these, encoding evolution and generation evolution both belong to input-bearing evolution, while autonomous evolution belongs to input-free evolution. All sequences start from the zero tensor and are wholly reset at the sequence boundary.

D.2 READ-OUT LAYER: PREDICTION READ-OUT

The read-out layer draws prediction directly from the evolutionary state of the state network; the read-out map is realized by the output projection.

First, pooled features of five node subsets are extracted by the $\mathrm{MeanPool}$ operator:

$$o = \mathrm{MeanPool}(O;\mathbf{S}), \quad i = \mathrm{MeanPool}(I;\mathbf{S}), \quad c = \mathrm{MeanPool}(C;\mathbf{S}),$$
$$f = \mathrm{MeanPool}(F;\mathbf{S}), \quad m = \mathrm{MeanPool}(M;\mathbf{S}),$$

where $F = I \cup W \cup O$ is the read-out auxiliary region.

Second, the full read-out feature is obtained by superposing the base path, the structural residual, and the slot alignment, with the semantic residual merged into the base path. All parts come from the same evolutionary state. The construction follows:

- **Semantic residual**: using the original token projection vectors of the input window $H_t = (e_{t-|I|+1}, \ldots, e_t)$, it provides a directly differentiable path to E_{in} for training:

  $$q_{\text{sem}} = [o; i; c; e_t; \mathrm{mean}(e_{t-3:t}); \mathrm{mean}(H_t)], \qquad u_{\text{sem}} = F_{\text{sem}}(\mathrm{LN}(q_{\text{sem}})),$$
  $$\delta_{\text{sem}} = \mathrm{Bounded}\left(u_{\text{sem}}, u_{\text{sem}}, \sigma(\gamma_{\text{sem}})c_{\text{read}}c_{\text{sem}}^{\max}\right).$$

  Here c_{read} is a scalar hyperparameter controlling the amplitude of the semantic residual, and together with $c_{\text{sem}}^{\max}$ bounds the residual scale, lest it disturb the base read-out path. F_{sem} is a two-layer fully connected map with input dimension $5d + d = 6d$, hidden dimension $2d$, output dimension d, and GELU activation in the hidden layer.
- **Base path**: the pooled features are fused into the base read-out feature r_{base}, with a gating mechanism dynamically weighting features from different sources:

  $$r_{\text{base}} = \mathrm{LN}\left(r_0 + g_{fs} \odot W_f f + (1 - g_{fs}) \odot W_c c + \delta_{\text{sem}}\right)$$

  Here $r_0 = W_O o + b_O + g_I \odot (W_i i + b_i)$, and g_I, g_{fs} are gate weights generated by the Gate operator, with

  $$g_I = \mathrm{Gate}([o; i; c]; W_I, b_I), \qquad g_{fs} = \mathrm{Gate}([f; c; r_0]; W_{fs}, b_{fs}),$$

  Gate being as defined in Eq. (6).
- **Structural residual**: The historical observation feature μ_u is given by the sequence runtime, and is a different variable from the compatible measurement history $\mathcal{Q}_u$ in the controller input. Here e_u^{pred} and e_u^{corr} are respectively the prediction error and the correction error of the most recent matured observation:

  $$\mu_u = [e_u^{\text{pred}}, e_u^{\text{corr}}, \mathrm{RMS}(v_u), \mathrm{RMS}(a_u), \|\Delta\boldsymbol{\phi}_u\|, \|\delta\mathbf{S}_u\|_{\mathrm{RMS}}, B_u^m/B_0^m, B_u^w/B_0^w] \in \mathbb{R}^8.$$

  Using the learned output node weights a_m (satisfying $\sum_l a_{m,l} = 0$ and $\|a_m\|_2 = 1$) and the historical observation feature μ_u, a structural bias is supplied:

  $$z_m = \sum_l a_{m,l} O_l, \quad \omega = \mathrm{softmax}(W_\omega[o; \mathbf{S}_E; m; \mu_u] + b_\omega), \quad u_{\text{str}} = W_\uparrow \sum_s \omega_s \,\mathrm{GELU}(W_\downarrow \mathrm{LN}(q_s)),$$

  $$\delta_{\text{str}} = \mathrm{Bounded}\left(u_{\text{str}}, r_{\text{base}}, \sigma(\gamma_{\text{str}})c_{\text{read}}\kappa_{\text{str}}\right),$$

  where $q_s \in \{z_1, \ldots, z_R, \mathbf{S}_E, m\}$. $z_1, \ldots, z_R$ are pooled-subset features of the output region O and the auxiliary region F, with $R = 4$, corresponding respectively to linear projections of the four pooled vectors O, I, F, M; their purpose is to give the structural residual several reference directions. a_m is derived from an unconstrained parameter $\tilde{a}_m \in \mathbb{R}^{|O|}$: $a_m = \tilde{a}_m - \mathrm{mean}(\tilde{a}_m)$, then divided by $\|\tilde{a}_m - \mathrm{mean}(\tilde{a}_m)\|_2$.
- **Slot alignment**: for the generation position $k_j = j - 1$, a bias term between the output node and the pooled feature is supplied:

  $$\delta_{\text{slot},k_j} = \mathrm{Bounded}\left(\sigma(\gamma_{\text{slot}})(W_{\text{slot}}\,\mathrm{LN}(O_l - o) + b_{\text{slot}}), r_{\text{base}} + \delta_{\text{str}}, \kappa_{\text{slot}}\right)$$

  where O_l is the feature of the $l = k_j \bmod |O|$-th node in the output region.

The final logits and probabilities are obtained from the base feature plus the two residuals:

$$r_j = r_{\text{base}} + \delta_{\text{str}} + \delta_{\text{slot},k_j}, \quad \ell_j = W_{\text{out}} r_j + b_{\text{out}}, \quad p_j = \mathrm{softmax}(\ell_j),$$

where $W_{\text{out}}, b_{\text{out}}$ are independent output projection parameters (belonging to the slow parameters $\boldsymbol{\theta}$).

D.3 Decision Layer: The Controller

The MetaPred model comprises two independent neural networks: one is the state network realized by the SiFu graph, the other the controller network executing meta-state control. The state network is organized by the five node partitions already defined; nodes communicate by node-axis mixing operators, and within a node the channel transform handles processing. Its parameters are the slow parameters $\boldsymbol{\theta}$ and the fast parameters $\boldsymbol{\phi}_u$. The state network interacts with the outside through the input projection matrix E_{in} and the output projection matrix W_{out}.

The controller network is a feedforward policy-value network with parameters independent of the state network, in an actor-critic structure. It receives the causal state summary o_u, and after one learnable layer normalization and two fully connected hidden layers obtains a shared representation of the same dimension:

$$z_u = \tanh(W_2 \tanh(W_1 \,\mathrm{LN}(o_u) + b_1) + b_2)\,,$$

where LN is a learnable layer normalization. The shared representation z_u connects simultaneously to six action heads and one scalar value head. The value head emits a scalar estimate $V_{\boldsymbol{\psi}}(o_u)$, used for PPO training.

The controller network is invoked once before each executed evolutionary step begins. From the state tensor and auxiliary metadata it reads the causal state summary o_u, emits a set of measurement and update actions a_u, and thereby configures the runtime conditions χ_u upon which subsequent evolution depends. The action configures only the runtime conditions χ_u; the state tensor and the evolutionary-step counter hold their values. The controller's scheduling of the self-measurement sub-process shows itself in the measurement actions a_u^{tier}, a_u^{focus}, a_u^{corr}: a_u^{tier} decides whether to execute direct or simulated measurement this time and at what simulation depth; a_u^{focus} designates the focus of measurement; a_u^{corr} decides the strength of state correction. Thus the second-order recursive dynamical sub-process is managed in one hand by the controller, not left to a fixed schedule. The timing and triggering condition of the control parameter $\boldsymbol{\psi}$ update are given by the PPO step determined in the training algorithm of the model.

The controller's input and output data flow is as follows:

The controller reads the causal state summary o_u, containing only information available from executed steps:

$$o_u = [s_c(\mathbf{S}_u);\, r_u;\, \mathcal{Q}_u;\, \tau_u;\, \zeta_u;\, a_{u-1};\, \mathrm{stage}_u;\, b_u;\, d_u]$$

where

- $s_c(\mathbf{S}_u)$: the control state summary, containing the global mean, RMS, RMS after removing the mean along the node axis, the respective RMS of input/output/internal/self-coordination nodes, and the finite-value ratio;
- r_u: the most recent running signal, containing prediction error, intrinsic energy, RMS of velocity and acceleration, the norms of fast-parameter update and state correction, the last reward, and the parameter version;
- $\mathcal{Q}_u$: the compatible measurement history, containing differential statistics of existing compatible measurements and the corresponding validity information;
- τ_u: the normalized number of steps since the last measurement for the nine classes of measurement objects;
- ζ_u: whether the nine classes of measurement objects have no record yet;
- a_{u-1}: the indices of the six actions from the last time;
- stage_u: the one-hot encoding of the invocation stage (encoding, autonomous, generation);
- b_u: the remaining ratios of the measurement budget and the fast-parameter update budget;
- d_u: running diagnostics such as phase, sample-pool and prediction-queue occupancy, first- and second-order differences of energy and their validity, self-measurement residual and loss, and calibration state.

All components fed into the controller have their gradients stopped.

The controller emits the action a_u, obtained by sampling from six discrete action heads, written as

$$a_u = \left(a_u^{\text{tier}}, a_u^{\text{focus}}, a_u^{\text{upd}}, a_u^{\text{str}}, a_u^{\text{corr}}, a_u^{\text{ratio}}\right),$$

where a_u^{tier} is the measurement tier, a_u^{focus} the measurement object, a_u^{upd} the fast-parameter update switch, a_u^{str} the fast-parameter update strength, a_u^{corr} the state-correction strength, and a_u^{ratio} the sampling ratio of matured observations for updating fast parameters. All components are discrete tiers; the six action heads each produce a categorical distribution, and after masking illegal categories, sampling is done according to the distribution to form the action a_u. The following situations further override the sampling result and execute the default action: insufficient coverage, warm-up phase, budget exhaustion, uncalibrated direct-measurement parameters, and simulated-measurement interval. These override rules are hard implementation constraints, not part of the control-parameter learning. When the update switch is off, the index of the sampling-ratio action head is set to zero; requesting an update and actually executing it are two distinct steps.

The controller is invoked under all three running modes (see Table 4); only TRAIN and ADAPTIVE allow fast-parameter update and state correction; under FROZEN the controller still computes actions, but their effect is confined to configuring runtime conditions, and the control parameters ψ are frozen under all evaluation modes.

E METAPRED TRAINING ALGORITHM: FULL SPECIFICATION

This section describes the update layer in MetaPred's six-layer structure, namely the learning correction of the self-measurement process. The three parameter families are updated separately by time scale: the slow parameters $\boldsymbol{\theta}$ are shared across sequences and updated by batch, the fast parameters $\boldsymbol{\phi}_u$ are sequence-local and updated by in-sequence events, and the control parameters $\boldsymbol{\psi}$ are shared across sequences and updated along the policy trajectory.

MetaPred's organization of training signals differs from layer-by-layer backpropagation in the traditional neural-network sense. The latter implies three premises: forward and backward share the same layer structure; the loss is a single terminal function; and within a single weight-update cycle, the backward computation unfolds layer by layer along the layer order, and this path is unique. MetaPred adopts a different organization on all three premises. Gradient computation itself still uses standard automatic differentiation; the difference lies in the source of the training signals and the organization of the parameters.

First, the SiFu graph is divided without hierarchy, and the parameters are partitioned by functional role.

Second, the training signal consists of multiple self-comparisons of the same state from different views: $\mathcal{L}_{\text{pred}}$ compares the state with its successor, $\mathcal{L}_{\text{fp}}$ compares the state with a one-step replay, $\mathcal{L}_{\text{self}}$ compares a one-step simulation with a K-step simulation, and $\mathcal{L}_{\text{direct-self}}$ compares the predicted residual with the simulated residual.

Third, the weight updates for the main body of the MetaPred model (the state network) are completed synchronously in a single pass per update cycle: gradients are computed once over the state tensor by automatic differentiation, and then the functionally grouped parameters are updated synchronously in one pass. Unlike traditional hierarchical neural networks, there is no backward propagation path that unfolds along layer order; instead, the backward paths from multiple observation points converge on the same hierarchy-free graph, rather than being propagated layer by layer.

This form is a necessary choice under the AGI mathematical theory. The single-state constraint requires the path of encoder-decoder unification, so that representation and prediction share the realization mechanism and parameters; the endogenous-supervision constraint requires the loss to come from comparison of the state with itself; the training–inference coupling constraint requires the training path to exist equally at inference; and the parameter-functional-heterogeneity constraint requires general dynamics, sequence-local adaptation, and resource-allocation policy to be updated separately at their own time scales. Together, the four constraints yield a unique workable form: multiple self-comparisons upon a single state, parameter update grouped by function, and training and inference sharing one evolutionary skeleton. This derivation chain is coherent: the six-layer decomposition of the architecture is determined by the functional requirements of the MetaPred primitive; the three-way separation of the training algorithm is determined by the heterogeneity of

parameter time scales; and the two designs are unified through the same dynamical skeleton. The training and inference flows described below are all realized upon the standard dynamical routine.

E.1 Multi-task Training Objective

The training objective consists of the external task loss and the endogenous self-supervised loss, all endogenous self-supervised losses calling the RelMSE operator:

$$\boxed{\mathcal{L}_{\text{outer}} = \mathcal{L}_{\text{task}} + \lambda_{\text{int}} \left(\lambda_p \mathcal{L}_{\text{pred}} + \lambda_f \mathcal{L}_{\text{fp}} + \lambda_d \mathcal{L}_{\text{dyn}} + \lambda_c \mathcal{L}_{\text{direct-self}}\right) + \lambda_s \mathcal{L}_{\text{self}}} \tag{16}$$

where the per-observation state prediction loss $\mathcal{L}_{\text{pred},i}$ is defined below under the state prediction loss term; since the target $\mathbf{S}_i^{\text{pre}}$ stops gradient, its gradient returns only through the state predictors $p_{\boldsymbol{\theta}}, p_{\boldsymbol{\phi}_u}$. Let $\mathcal{D}_{\text{pred}}$ be the set of matured state-prediction observations, $\mathcal{D}_{\text{fp}}$ the set of observations that have reached the pre-correction state, and $\mathcal{D}_{\text{dyn}}$ the set of observations with available dynamical trends. The definitions and uses of each loss term follow:

- **External task loss**: $\mathcal{L}_{\text{task}} = -\frac{1}{BH} \sum_{b=1}^{B} \sum_{j=1}^{H} \log p_{b,j}[y_{b,j}]$, the next-token cross-entropy, available only when external labels exist.
- **State prediction loss**: the per-observation loss is

  $$\mathcal{L}_{\text{pred},i} = \mathrm{RelMSE}(\widehat{\mathbf{S}}_i^{\text{pre}}, \mathbf{S}_i^{\text{pre}}),$$

  and the overall loss is

  $$\mathcal{L}_{\text{pred}} = \mathbb{E}_{i \in \mathcal{D}_{\text{pred}}} \mathcal{L}_{\text{pred},i},$$

  monitoring the state predictors' accuracy upon the true one-step successor, with the target $\mathbf{S}_i^{\text{pre}}$ stopping gradient.
- **Fixed-point proxy loss**:

  $$\mathcal{L}_{\text{fp}} = \frac{1}{|\mathcal{D}_{\text{fp}}|} \sum_{i \in \mathcal{D}_{\text{fp}}} \mathrm{RelMSE}\left(\Phi_{q_i}^{\text{step}}(\mathbf{S}_i^{\text{pre}}; \chi_i), \mathbf{S}_i^{\text{pre}}\right),$$

  replaying one ordinary evolutionary call under the stored parameter-group index q_i and historical condition χ_i, to probe whether the actually reached pre-correction state remains unchanged under one-step evolution; it does not advance the true evolutionary-step counter. This probe differs from simulated measurement in that it acts upon the **already reached** pre-correction state, whereas the two paths of simulated measurement both start from the same **current** state.
- **Dynamical trend loss**: for each matured observation i, define

  $$\mathcal{L}_{\text{dyn},i} = \frac{w_s \mathcal{L}_{\text{state},i} + I_i^m w_m \mathcal{L}_{\text{measurement},i} + I_i^e w_e \mathcal{L}_{\text{energy},i}}{w_s + I_i^m w_m + I_i^e w_e},$$

  and take $\mathcal{L}_{\text{dyn}} = \mathbb{E}_{i \in \mathcal{D}_{\text{dyn}}} \mathcal{L}_{\text{dyn},i}$, supervising consistency among state trend, measurement-statistic trend, and energy trend, computed only from compatible predecessors of matured observations. Here $I_i^m, I_i^e \in \{0, 1\}$ are indicators respectively for "the measurement trend of observation i is available" and "the energy trend is available"; $\mathcal{L}_{\text{state}}, \mathcal{L}_{\text{measurement}}, \mathcal{L}_{\text{energy}}$ are the RelMSE losses of state trend, measurement-statistic trend, and energy trend respectively; $\mathcal{D}_{\text{dyn}}$ is the set of observations with available dynamical trends;

  Let $s(\mathbf{S})$ be the state summary, and u_i^0, u_i^1 the issue and target evolutionary-step indices of observation i, with $\Delta u_i = u_i^1 - u_i^0 \geq 1$; define the h-th order trend

  $$v_i^{(h)} = \frac{s^{(h)}(\mathbf{S}_i^{\text{pre}}) - s^{(h)}(\mathbf{S}_i)}{\Delta u_i}, \qquad h \in \{1, 2\}, \tag{17}$$

  where $s^{(1)} = s$, and $s^{(2)}$ is the difference of the first-order trend along the evolutionary step, i.e., the second-order trend (acceleration). The state trend head $T_{\boldsymbol{\theta}}$ takes as input the state summary $s(\mathbf{S}_i) \in \mathbb{R}^{d_h}$ at the time the observation was emitted, and simultaneously outputs a first-order trend correction and a second-order trend prediction:

  $$\left[b_i^{(1)}, \widehat{v}_i^{(2)}\right] = T_{\boldsymbol{\theta}}(s(\mathbf{S}_i)) \in \mathbb{R}^{2d_h},$$

where both output components lie in $\mathbb{R}^{d_h}$. Let the hidden dimension of this trend head be h_T, with $h_T \geq 2d_h$; its structure is

$$\mathrm{LN} \to \mathrm{Linear}(d_h, h_T) \to \mathrm{GELU} \to \mathrm{Linear}(h_T, 2d_h).$$

The first-order trend prediction is obtained by adding the rate of change implied by the state prediction to the correction from the trend head:

$$\widehat{v}_i^{(1)} = \frac{s(\widehat{\mathbf{S}}_i^{\mathrm{pre}}) - s(\mathbf{S}_i)}{\Delta u_i} + b_i^{(1)}.$$

The second-order trend prediction $\widehat{v}_i^{(2)}$ is given directly by the trend head. The per-observation state trend loss is

$$\mathcal{L}_{\mathrm{state},i} = \frac{\sum_{h \in \mathcal{A}_i^v} \mathrm{RelMSE}(\widehat{v}_i^{(h)}, v_i^{(h)})}{|\mathcal{A}_i^v|}, \tag{18}$$

where $\mathcal{A}_i^v \subseteq \{1, 2\}$ is the set of available orders; $\widehat{v}_i^{(1)}$ is the combined prediction above, and $\widehat{v}_i^{(2)}$ is the second-order prediction from the trend head. The targets at each order stop gradient, and the supervision loss is computed only for valid matured observations.

The input to the measurement trend head is formed by concatenating the d_h-dimensional state summary with the one-hot encodings of the nine classes of measurement objects, giving input dimension d_h+9; its hidden dimension is denoted h_m, and its output dimension is $2c_m$, where c_m is the dimension of the measurement-statistic vector. Its output is split into two c_m-dimensional vectors, predicting the first- and second-order trends of the measurement statistics respectively. The energy trend head has input dimension d_h, hidden dimension h_e, and output dimension 2, predicting the first- and second-order trends of the scalar energy respectively. Both trend heads use layer normalization, two linear maps with an intermediate GELU activation, and compute the relative mean-squared error only over the valid orders.

The weights are taken as $(w_s, w_m, w_e) = (1.0, 0.01, 0.1)$. The concrete construction of the three sub-losses uses the $\mathrm{MeasVec}$ and Trend operators.

- **Direct-measurement parameter loss**: $\mathcal{L}_{\text{direct-self}}$ is defined by Eq. (15), calibrating the direct-measurement parameters' one-step state prediction and two-step residual prediction accuracy.
- **Self-measurement loss**: $\mathcal{L}_{\text{self}}$ is defined by Eq. (11), the endogenous fixed-point self-supervision signal.

The loss weights are set as: $\lambda_{\text{int}} = 0.01$, $(\lambda_p, \lambda_f, \lambda_d) = (1, 0.1, 0.5)$, $\lambda_c = 1$, $\lambda_s = 0.1$.

The aforementioned loss terms share one and the same per-observation record. The transition observation produced at each executed evolutionary step is written as

$$\mathcal{R}_i = (\mathbf{S}_i, \mathbf{S}_i^{\mathrm{pre}}, s(\mathbf{S}_i), v_i, a_i, \mathrm{stage}_i, q_i, \nu_i, \mathcal{K}_i, \mathbf{S}_{\mathrm{pred},i}, \Delta_{\mathrm{self},i}^{(2)}),$$

where $s(\mathbf{S}_i)$ is the state summary (defined by Eq. (9)), v_i the trend, a_i the controller action, stage_i the invocation stage, q_i the parameter-group index, ν_i the fast-parameter version, and $\mathcal{K}_i$ the clipped input and historical observations; the last two fields exist only for $K = 2$ simulated measurements. This record is the common input for direct-measurement parameter calibration, slow-update preview, and PPO policy update.

E.2 Separate Update Rules

The learnable parameters fall into three families by time scale: the slow parameters $\boldsymbol{\theta}$ are the cross-sequence parameters of the state network, the fast parameters $\boldsymbol{\phi}_u$ the sequence-local parameters of the state network, and the control parameters $\boldsymbol{\psi}$ the parameters of the controller network. The three families differ essentially in time scale: $\boldsymbol{\theta}$ is updated by batch, $\boldsymbol{\phi}_u$ by in-sequence events, and $\boldsymbol{\psi}$ by policy trajectory. This heterogeneity makes a unified update mechanism impossible; separation is mandatory. Should the gradient paths of the three families contaminate one another, the endogenous self-supervision signal would spoil the general representation, or the controller would exploit self-measurement to cheat. This isolation is a grouping at the optimization level: MetaPred's state $\mathbf{S}_{t,u}$

remains a single global tensor, and all parameters within $\boldsymbol{\theta}$ are updated synchronously in one update, corresponding to the hierarchy-free dynamical weights upon the SiFu graph.

The three families differ in sharing scope: $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ are shared across sequences, while $\boldsymbol{\phi}_u$ is sequence-local. Their coverage and update mechanisms appear in Table 5.

Table 5: The three parameter families: coverage and update mechanism

Parameter family	Coverage	Update source	Trigger condition
Slow parameters $\boldsymbol{\theta}$	Input projection matrix E_{in}, output projection matrix $W_{\text{out}}, b_{\text{out}}$; slot projection vectors $\{s_k, p_k\}$; modal reference vectors $\{\mathbf{m}_k\}$; all weights of the five single-step-evolution stages; read-out path weights; slow prediction parameters $p_{\boldsymbol{\theta}}$ and slow correction parameters $c_{\boldsymbol{\theta}}$	Direct gradient upon $\mathcal{L}_{\text{outer}}$ (AdamW)	Slow-parameter update boundary s
Fast parameters $\boldsymbol{\phi}_u$	Fast prediction parameters $p_{\boldsymbol{\phi}_u}$, fast correction parameters $c_{\boldsymbol{\phi}_u}$	Direct gradient upon $\mathcal{L}_{\text{inner}}$ (clipped SGD)	Every request by the controller within a sequence
Control parameters $\boldsymbol{\psi}$	All parameters of the controller network, including the shared encoder, six action heads, and the value head	PPO policy gradient (independent Adam optimizer)	When the policy trajectory matures

The update sources of the three families fall into two kinds. The first is the direct gradient: $\boldsymbol{\theta}$ corresponds to $\mathcal{L}_{\text{outer}}$, $\boldsymbol{\phi}_u$ to $\mathcal{L}_{\text{inner}}$, both being gradients upon a fixed loss function. The second is the policy gradient: the update of $\boldsymbol{\psi}$ comes from PPO, whose objective $\mathcal{L}_{\text{PPO}}$ depends upon sampled trajectories and delayed advantage estimates, belonging to the policy-gradient source. The two sources determine the differences in update frequency and optimizer form.

- **Slow parameter $\boldsymbol{\theta}$ update**: using the AdamW optimizer, executed at the slow-parameter update boundary s:

$$\bar{g}_s^{\boldsymbol{\theta}} = \text{ClipNorm}(g_s^{\boldsymbol{\theta}}, c_{\boldsymbol{\theta}}), \quad \boldsymbol{\theta}_{s+1} = \boldsymbol{\theta}_s + a_s \cdot \text{AdamW}(\boldsymbol{\theta}_s, \bar{g}_s^{\boldsymbol{\theta}}; \alpha_s, \beta_1, \beta_2, \lambda_w)$$

  where $g_s^{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}_s} \overline{\mathcal{L}}_s$ is the accumulated batch gradient, a_s an alternative scaling factor (1 for ordinary steps, 0.25/0.5/1 for preview steps), and $c_{\boldsymbol{\theta}}$ the global gradient-norm cap.

  Optionally enable the **slow-update preview mechanism**: before updating, construct alternative parameters with alternative scaling factors $a \in \{1, 0.5, 0.25\}$, compute the joint objective J_a from the same replayed training fragment, and choose the alternative with the smallest J_a satisfying $J_a \leq J_0 + \tau(J_0)$ (with $\tau(z) = \epsilon_{\text{abs}} + \epsilon_{\text{rel}}|z|$ as tolerance); if no alternative qualifies, keep the original parameters.

  Slow parameters are updated synchronously at batch boundaries, because $\boldsymbol{\theta}$ covers the entire SiFu graph and optimization proceeds by global parameter group; there is no hierarchical structure that could be backpropagated layer by layer.

- **Fast parameter $\boldsymbol{\phi}_u$ update**: using the clipped SGD optimizer, executed when the controller enables fast-parameter update:

$$g_u^{\boldsymbol{\phi}} = \nabla_{\boldsymbol{\phi}_u} \mathcal{L}_{\text{inner}}, \quad \boldsymbol{\phi}_u^+ = \boldsymbol{\phi}_u - \alpha_{\boldsymbol{\phi}} a_u^{\text{str}} \, \text{ClipNorm}(g_u^{\boldsymbol{\phi}}, c_{\boldsymbol{\phi}}),$$

  Let $\mathcal{M}_u$ be the set of observations selected by the controller that matured before evolutionary step u:

$$\mathcal{L}_{\text{inner}} = \mathbb{E}_{i \in \mathcal{M}_u} \left[\lambda_p \mathcal{L}_{\text{pred},i} + \lambda_d^{\text{fast}} \mathcal{L}_{\text{state},i} + \lambda_c \mathcal{L}_{\text{direct-self},i} \right],$$

  The inner update comprises state prediction, state trend, and direct-measurement losses; a_u^{str} is the update strength chosen by the controller. Here $\mathcal{L}_{\text{state},i}$ is the per-observation state trend loss, constructed identically to $\mathcal{L}_{\text{state}}$. The inner-update weight is $\lambda_d^{\text{fast}} = 0.001$.

  Fast parameters use clipped SGD because their update scope is confined to the current sequence, and clipping prevents a single update from being too strong and disrupting the state trajectory.

- **Control parameter ψ update**: using the PPO algorithm, delayed until a matured policy trajectory is ready:

  Reward function: $R_u = \Delta E_{\text{direct}} + \Delta E_{\text{corr}} - \beta_c C_u - \beta_w \|\Delta \boldsymbol{\phi}_u\|^2 - \beta_s \|\delta \mathbf{S}_u\|^2_{\text{RMS}} - \beta_\infty \mathbf{1}[\mathbf{S}_{u+1} \text{ contains non-finite values}]$, where ΔE_{direct} is the improvement of direct-measurement parameters upon independent held-out observations, ΔE_{corr} the improvement of self-consistency under fixed conditions, and C_u the accumulated cost of measurement/update/correction.

  Define the direct-measurement parameter energy upon held-out observations:

  $$E_{\text{direct}}(\boldsymbol{\phi}; r) = \lambda_p \mathcal{L}_{\text{pred},r}(\boldsymbol{\phi}) + \lambda_d^{\text{fast}} \mathcal{L}_{\text{state},r}(\boldsymbol{\phi}) + \lambda_c \mathcal{L}_{\text{direct-self},r}(\boldsymbol{\phi}), \tag{19}$$

  where r is an independent held-out observation; then $\Delta E_{\text{direct}} = E_{\text{direct}}(\boldsymbol{\phi}_u; r_u) - E_{\text{direct}}(\boldsymbol{\phi}_u^+; r_u)$. Define the fixed-condition self-consistency at a configuration-probe call:

  $$F_q(X) = \text{RelMSE}\left(\Phi_q^{\text{step}}(X; \chi_u^{\text{real}})_C, X_C\right), \tag{20}$$

  where the subscript C means taking the sub-tensor of the coordination region $C = \{E\} \cup M$; then $\Delta E_{\text{corr}} = F_q(\mathbf{S}_{u+1}^{\text{pre}}) - F_q(\mathbf{S}_{u+1})$. Both evaluations keep the selected parameter-group index and do not advance the true call counter.

  Advantage estimation: the generalized advantage estimate (GAE) is used to compute the advantage function A_u, with target value $G_u = A_u + \bar{V}_u$.

  PPO loss: $\mathcal{L}_{\text{PPO}} = -\mathbb{E}_u\left[\min(\rho_u \widetilde{A}_u, \bar{\rho}_u \widetilde{A}_u)\right] + c_V \mathbb{E}_u (V_{\boldsymbol{\psi}}(o_u) - G_u)^2 - c_H \mathbb{E}_u \mathcal{H}(\pi_{\boldsymbol{\psi}}(\cdot|o_u))$, where $\rho_u = \exp(\log \pi_{\boldsymbol{\psi}}(a_u|o_u) - \bar{\ell}_u)$ is the policy probability ratio, $\bar{\ell}_u$ the log-probability of the behavior policy at action a_u, and $\bar{\rho}_u = \text{ClipTo}(\rho_u, 0.8, 1.2)$ the clipped probability ratio.

  The control parameters $\boldsymbol{\psi}$ are randomly initialized and trained from scratch, independent of other parameters.

  The controller can only be trained by reinforcement learning akin to PPO, because its objective is the long-term return of resource allocation, which can be evaluated only several evolutionary steps after the action, and cannot be obtained by taking direct gradients of a single-step loss. PPO updates only $\boldsymbol{\psi}$; $\boldsymbol{\theta}$ and $\boldsymbol{\phi}_u$ are updated by their respective optimizers.

F Conditional Convergence and Residual Decomposition Proofs

This section provides the theoretical basis of MetaPred's self-measurement mechanism, including the telescoping decomposition property of the residual and the conditional convergence theorem. These conclusions provide a conditional mathematical guarantee for endogenous self-supervision; the convergence of the model state during training is determined by concrete parameters.

F.1 Telescoping Decomposition of the Residual

For brevity, the following proofs omit the condition parameter χ of single-step evolution, writing $\Phi_q^{\text{step}}(\mathbf{S}) := \Phi_q^{\text{step}}(\mathbf{S}; \chi)$.

Consider the K-step simulation starting from executed evolutionary step u: $\mathbf{S}_{\text{sim}}^{(K)} = \Phi_q^{(K)}(\mathbf{S}_u)$. By definition, $\Phi_q^{(K)}$ is the composite of K single-step evolutions. Define the single-step residual as:

$$\Delta_u^{(1)} = \Phi_q^{\text{step}}(\mathbf{S}_u) - \mathbf{S}_u$$

Then the K-step residual $\Delta_{\text{self}}^{(K)}(u) = \mathbf{S}_{\text{sim}}^{(K)} - \mathbf{S}_{\text{pred}}$ (where $\mathbf{S}_{\text{pred}} = \Phi_q^{\text{step}}(\mathbf{S}_u)$) satisfies the following decomposition:

定理 **1** (Residual decomposition)**.** *For $K \geq 2$, the K-step self-measurement residual can be decomposed as the cumulative sum of the first $K-1$ single-step residuals:*

$$\Delta_{self}^{(K)}(u) = \sum_{k=1}^{K-1} \Delta_{u+k}^{(1)}$$

where $\Delta_{u+k}^{(1)}$ is the single-step residual produced at the k-th step of the simulation path.

Proof. By the definition of the composite operator:

$$\begin{aligned}\mathbf{S}_{\text{sim}}^{(K)} &= \Phi_{q+K-1}^{\text{step}} \circ \cdots \circ \Phi_{q+1}^{\text{step}} \circ \Phi_{q}^{\text{step}}(\mathbf{S}_u) \\ &= \Phi_{q+K-1}^{\text{step}} \circ \cdots \circ \Phi_{q+1}^{\text{step}}(\mathbf{S}_{\text{pred}})\end{aligned}$$

Let $\mathbf{S}_{\text{sim}}^{(k)}$ be the k-th simulated state; then:

$$\begin{aligned}\Delta_{\text{self}}^{(K)}(u) &= \mathbf{S}_{\text{sim}}^{(K)} - \mathbf{S}_{\text{pred}} \\ &= \sum_{k=1}^{K-1} \left(\mathbf{S}_{\text{sim}}^{(k+1)} - \mathbf{S}_{\text{sim}}^{(k)}\right) \\ &= \sum_{k=1}^{K-1} \Delta_{u+k}^{(1)}\end{aligned}$$

where $\mathbf{S}_{\text{sim}}^{(1)} = \mathbf{S}_{\text{pred}}$. □

This decomposition shows that the K-step simulation ($K > 1$) captures the cumulative trend of the dynamics over several evolutionary steps, rather than a single-step residual signal; this provides the theoretical basis for the trend supervision $\mathcal{L}_{\text{dyn}}$.

F.2 Conditional Convergence Theorem

MetaPred's self-measurement mechanism borrows the idea of Lambek's fixed-point lemma (Lambek, 1968), but guarantees convergence only under certain conditions. We consider the full parameter-group cycle $\mathcal{C}_{q+1} = \Phi_{q+1}^{(Q)}$, depending upon the slow parameters $\boldsymbol{\theta}$ and the runtime condition χ.

定理 **2** (Conditional convergence)**.** *Assume that on a closed subset $\mathcal{X} \subseteq \mathbb{R}^{N\times d}$, the full parameter-group cycle $\mathcal{C}_{q+1}$ is Lipschitz continuous with Lipschitz constant $L < 1$ (i.e., a contraction). Then by the Banach fixed-point theorem there exists a unique fixed point $\mathbf{S}^* \in \mathcal{X}$ such that $\mathcal{C}_{q+1}(\mathbf{S}^*) = \mathbf{S}^*$, and for any initial state $\mathbf{S}_0 \in \mathcal{X}$, the iteration sequence $\mathbf{S}_{n+1} = \mathcal{C}_{q+1}(\mathbf{S}_n)$ converges linearly to $\mathbf{S}^*$.*

Proof. Standard application of the Banach fixed-point theorem: a contraction upon a complete metric space has a unique fixed point. Here the metric is induced by the Frobenius norm, and $\mathcal{X}$ is a closed subset (a complete subspace). □

Important note: the above theorem is a conditional result. During training, the slow parameters $\boldsymbol{\theta}$ are continually updated, and the Lipschitz constant L of the parameter-group cycle $\mathcal{C}_{q+1}$ may exceed 1; convergence of the trained model to a fixed point is a conditional conclusion. The self-measurement loss $\mathcal{L}_{\text{self}}$ encourages the model to exhibit approximately contractive behavior locally; global convergence is determined by concrete parameters.

F.3 Geometric Property of the Correction Term in Self-Measurement

The correction term of self-measurement $\delta\mathbf{S}_u = \eta_{\text{eff}}\widetilde{\Delta}_{\text{self}}^{(K)}(u)$ has a clear geometric interpretation:

命題 **1** (Segment constraint of the correction term)**.** *If the simulated residual is used, $\widetilde{\Delta}_{self}^{(K)}(u) = \Delta_{self}^{(K)}(u)$, and the effective strength $\eta_{eff} \in [0, 1]$, then the corrected state $\mathbf{S}_{u+1}$ lies upon the segment between the true successor $\mathbf{S}_{u+1}^{pre}$ and the K-step simulation endpoint $\mathbf{S}_{sim}^{(K)}$.*

Proof. By definition $\mathbf{S}_{u+1} = \mathbf{S}_{u+1}^{\text{pre}} + \eta_{\text{eff}}(\mathbf{S}_{\text{sim}}^{(K)} - \mathbf{S}_{\text{pred}})$. Since $\mathbf{S}_{u+1}^{\text{pre}} = \mathbf{S}_{\text{pred}}$, one has:

$$\mathbf{S}_{u+1} = (1 - \eta_{\text{eff}})\mathbf{S}_{\text{pred}} + \eta_{\text{eff}}\mathbf{S}_{\text{sim}}^{(K)}$$

When $\eta_{\text{eff}} \in [0, 1]$, the above is a convex combination of the two points, hence lies upon the segment. □

This property confines state correction to the range of the simulation endpoints, maintaining the stability of the dynamics.

G Implementation Details and Update Scope

This section summarizes the engineering implementation details of MetaPred.

G.1 Network Architecture and Initialization

- **Node and auxiliary module configuration**: Different experiments use different state network scales. The specific configuration of the main model reported in this paper is as follows, where N is the total number of nodes, d is the state dimension per node, r is the bottleneck dimension of the prediction and correction adapters, and d_h is the state summary dimension:

Experimental model	N	d	r	d_h
Chinese 13M	256	256	8	16
English 13M	96	132	8	16
Chinese 129M	256	928	32	32

  The above main models all adopt $|I| = 16$, $|O| = 16$, $|E| = 1$, and $|M| = 40$, and the number of working nodes is $|W| = N - |I| - |O| - |E| - |M|$. Node masking in structural ablations is otherwise performed according to the corresponding experimental definition.
- **Parameter groups**: total parameter-group count $Q = 4$, the parameter-group index corresponding to evolutionary step u being $q_u = u \bmod Q$; each parameter group contains independent $\mathrm{GateMix}$ weights (as defined in Eq. (1)), cyclically reused.
- **Initialization scheme**:
  - Input projection matrix E_{in}, output projection matrix W_{out}: cold start with Xavier uniform initialization.
  - Linear layer weights: Xavier uniform initialization; bias terms initialized to 0.
  - Layer normalization LN: initialized to identity mapping ($\gamma = 1, \beta = 0$).
  - Linear transforms in the core gated mixing: output-layer weights multiplied by a 0.1 scaling factor to prevent initial signal explosion; newly added auxiliary heads use independent initialization on the order of 10^{-3}.
  - Fast parameters $\boldsymbol{\phi}_u$: bias terms initialized to zero; down-projection weights use Xavier uniform initialization, up-projection weights use $\mathcal{N}(0, 10^{-3})$; randomly initialized, independent of slow parameters.
  - Control parameters $\boldsymbol{\psi}$: randomly initialized, independent of other model parameters, and trained from scratch.
- **Slot projection vectors**: for each window position k in the input window, two learnable slot projection vectors s_k and p_k ($k = 1, \ldots, |I|$) correspond respectively to the slot embedding table and the position embedding table in the realization. Both are learnable parameters in $\mathbb{R}^d$, initialized by a normal distribution $\mathcal{N}(0, 0.02^2)$; they are trained together with the slow parameters $\boldsymbol{\theta}$.

G.2 Optimizer and Gradient Clipping

All gradient updates follow the separate update rules already defined; the concrete configuration is as follows:

- **Slow-parameter optimizer**: AdamW, parameters $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay $\lambda_w = 0.01$; global gradient-norm cap $c_{\boldsymbol{\theta}} = 1.0$ (implemented by the $\mathrm{ClipNorm}$ operator).
- **Fast-parameter optimizer**: clipped SGD, learning rate $\alpha_{\boldsymbol{\phi}} = 0.05$, gradient-norm cap $c_{\boldsymbol{\phi}} = 1.0$; the update strength is dynamically scaled by the controller action a_u^{str}.
- **Control-parameter optimizer**: Adam, learning rate 3×10^{-4}, used to update the control parameters $\boldsymbol{\psi}$ by the PPO objective. Slow parameters $\boldsymbol{\theta}$ use AdamW, fast parameters $\boldsymbol{\phi}_u$ use clipped SGD. The controller network's structure and parameter count appear in Appendix G.3.

- **Gradient-stopping rules**: gradient boundaries are distinguished by use. Diagnostic simulated measurements, the historical observation summary μ_u, and all components of the controller input o_u have their gradients stopped by the $\mathrm{sg}(\cdot)$ operator; when supervision is recomputed, predictions and residuals are recomputed, retaining gradients of the slow prediction parameters and slow correction parameters; the direct measurement used for state correction during training is recomputed after fast-parameter update, and when outer gradients are enabled, the corresponding gradients are retained to connect to the next state; the reference signals of $K = 2$ and $K = Q + 1$ simulations stop gradient. Simulated measurements retain gradients under training mode (TRAIN) and train the shared dynamics through the self-measurement loss; under evaluation mode they do not participate in gradients.

G.3 CONTROLLER NETWORK IMPLEMENTATION DETAILS

The controller network consists of the following parts in order: first, a learnable layer normalization LN is applied to the causal state summary o_u; then two hidden layers, each of dimension 64 with Tanh activation:

$$z_u = \tanh(W_2 \tanh(W_1 \,\mathrm{LN}(o_u) + b_1) + b_2)\,,$$

where $W_1 \in \mathbb{R}^{64\times 66}$, $W_2 \in \mathbb{R}^{64\times 64}$. The shared representation z_u connects simultaneously to six action heads and one scalar value head. The six action heads respectively produce discrete distributions for the measurement tier, the measurement object, the fast-parameter update switch, the fast-parameter update strength, the state-correction strength, and the fast-parameter update sampling ratio; the category counts are $4, 9, 2, 3, 3, 3$ in order, totaling 24 categorical scores; the value head emits a scalar estimate $V_{\boldsymbol{\psi}}(o_u)$.

The causal state summary o_u is 66-dimensional, concatenated from the following nine parts:

- Control state summary, 8 dimensions ($s_c(\mathbf{S}_u)$): global mean, RMS, RMS after removing the mean along the node axis, respective RMS of input/output/internal/self-coordination nodes, finite-value ratio;
- Most recent running signal, 8 dimensions: prediction error, intrinsic energy, RMS of velocity and acceleration, norms of fast-parameter update and state correction, last reward, parameter version;
- Compatible measurement history, 8 dimensions ($\mathcal{Q}_u$): differential statistics of existing compatible measurements and corresponding validity information;
- Measurement age, 9 dimensions: the normalized number of steps since the last measurement for the nine classes of measurement objects;
- Missing flags, 9 dimensions (ζ_u): whether the nine classes of measurement objects have no record yet;
- Last action, 6 dimensions: the six action indices from last time;
- Stage flags, 3 dimensions: one-hot encoding of the invocation stage (encoding, autonomous, generation);
- Remaining budget, 2 dimensions: remaining ratios of the measurement budget and the fast-parameter update budget;
- Running diagnostics, 13 dimensions: phase, sample-pool and prediction-queue occupancy, first- and second-order differences of energy and their validity, self-measurement residual and loss, calibration state, etc.

Total $8+8+8+9+9+6+3+2+13 = 66$. When constructing the observation, some continuous statistics are first subjected to signed logarithmic compression $f(x) = \mathrm{sign}(x)\log(1+|x|)$, then passed through the learnable layer normalization inside the controller network.

The nine classes of measurement objects are: self-consistency, input nodes, output nodes, internal nodes, self-coordination node, direct-measurement parameters, fast parameters, state network, policy. The nine classes correspond in order to the nine positions of the causal state summary components τ_u and ζ_u, and are also the nine candidate categories of a_u^{focus}. a_u^{focus} designates the focus of measurement; the actual measurement scope is determined by the measurement tier: direct measurement covers

self-consistency and the focus object; short-range simulation also covers input, output, internal nodes and fast parameters; long-range simulation covers all nine classes, in which case a_u^{focus} does not affect the object set.

The category counts and values of the controller's action components are as follows:

- a_u^{tier}: 4 categories, skip measurement, direct measurement, short-range simulation, long-range simulation;
- a_u^{focus}: one of 9 registered measurement objects;
- a_u^{upd}: 2 categories, no update, request update;
- a_u^{str}: 3 tiers, $0.25, 0.5, 1.0$;
- a_u^{corr}: 3 tiers, $0, 0.25, 0.5$;
- a_u^{ratio}: 3 tiers, $25\%, 50\%, 100\%$.

When the update switch is off, the index of the sampling-ratio action head is set to zero; budget and available records likewise constrain execution; requesting an update and actually executing it are two distinct steps.

The control parameters $\boldsymbol{\psi}$ comprise the layer normalization, two hidden layers, six action heads, and one value head, totaling about 1.02×10^4 parameters. These parameters are trained by an independent Adam optimizer under the PPO objective; the slow parameters $\boldsymbol{\theta}$ use AdamW, and the fast parameters $\boldsymbol{\phi}_u$ use clipped SGD.

G.4 Controller and Self-Measurement Scheduling

- **Measurement tiers**: $a_u^{\text{tier}} \in \{\text{SKIP}, \text{DIRECT}, \text{SHORT}, \text{LONG}\}$ corresponds to four measurement tiers: skip measurement, direct measurement, short-range simulation, and long-range simulation (the latter two being subdivisions of simulated measurement by depth). Skip measurement corresponds to zero signal; direct measurement gives an approximate residual from the direct-measurement parameters without executing simulation; simulated measurement actually runs state simulation, with depth determined by the parameter K, where $K = 2$ is short-range simulation and $K = Q + 1$ is long-range simulation; the current realization takes $Q = 4$, i.e., $K = 5$.
- **Budget constraints**: the initial measurement budget B_0^m and update budget B_0^w are set according to experimental scale (the current realization takes $512/32$ or $4096/256$); different measurement tiers have different consumption, with values $(0, 1, 4, 12)$; the fast-parameter update consumption also depends on the number of sampled observations; once the budget is exhausted, the corresponding operation is automatically disabled.
- **Warm-up and override**: warm-up is counted by optimizer steps; for the first several steps, simulated measurement is preferred; when the direct-measurement parameter calibration error exceeds the threshold (0.5), correction driven by direct measurement is disabled, and simulated measurement may be forced; two long-range simulations must satisfy a minimum interval of 16.

G.5 Loss Weights and Hyperparameters

The multi-task weights in Eq. (16) are configured as follows:

- Total endogenous loss weight: $\lambda_{\text{int}} = 0.01$.
- Sub-loss weights: $(\lambda_p, \lambda_f, \lambda_d, \lambda_c, \lambda_s) = (1.0, 0.1, 0.5, 1.0, 0.1)$.
- Dynamical trend loss weights: $(w_s, w_m, w_e) = (1.0, 0.01, 0.1)$, corresponding respectively to state, measurement, and energy trends.
- PPO hyperparameters: GAE parameters $(\lambda_{\text{GAE}}, \gamma) = (0.95, 0.99)$; clipping range $\bar{\rho}_u \in [0.8, 1.2]$; value-function coefficient $c_V = 0.5$; entropy coefficient $c_H = 0.01$.
- Slow-update preview tolerance: $\epsilon_{\text{abs}} = 10^{-4}$, $\epsilon_{\text{rel}} = 10^{-3}$.

H Detailed Experimental Results

This section assesses the predictive performance of the realized BriLLM-MetaPred as a complete language model, and further examines the internal mechanism, fast-parameter adaptation, and larger-scale training.

H.1 Experimental Questions and Comparison Setup

Table 6 gives the purpose of each group of experiments. The continuous-execution mode is the main language-modeling comparison; the sparse-auxiliary mode serves the analysis of the internal mechanism. The main comparison scale used is 13M; in addition, a 129M model of the same scale as GPT-1 is trained, to evaluate larger-scale training and generation.

Table 6: Experimental overview.

Research question	**Experimental object**	**Main observation**
Language modeling performance	Continuous-execution 13M against same-size GPT baseline	Validation PPL
Internal mechanism role	Sparse-auxiliary 13M mechanism ablation	FROZEN-mode prediction loss
Scale training and generation	Sparse-auxiliary 129M, two window sizes	Training endpoint, validation metrics, generation samples

Training modes and evaluation modes. This section continues the naming of MetaPred's three running modes and training modes. Two training modes are involved: the **continuous-execution mode** preserves node state and fast parameters across fragments within a sequence; the **sparse-auxiliary mode** runs the self-measurement mechanism periodically alongside backbone training. Evaluation is conducted under the FROZEN and ADAPTIVE modes, differing only in whether fast-parameter update and state correction are allowed. Below, "scale" denotes parameter count (M for million), and "length" the training window length.

Evaluation metrics and reporting. Language modeling uses validation-set average negative log-likelihood (NLL), perplexity $\mathrm{PPL} = \exp(\mathrm{NLL})$, and bits-per-character for character tasks $\mathrm{BPC} = \mathrm{NLL}/\ln 2$, all lower being better. Three-seed experiments use 42, 43, 44, reporting seed-level mean and sample standard deviation. Teacher forcing scores the prediction at the current position first, then writes the reference token; ADAPTIVE's fast-parameter update uses internal feedback and does not use task scores as gradient targets.

Chinese data split. The Chinese experiments use the THUCNews-v3 dataset, split 90:5:5 by exact and conservative near-duplicate article groups, with train/validation/test of 728,267/40,382/40,533 articles; the vocabulary consists of the 6,512 tokens occurring at least 5 times in the training set. Full hyperparameters and optimizer configuration appear in Appendix G.

English data split. The English experiments use BookCorpusOpen. After exact whole-book SHA-256 deduplication, 17,785 books remain from the 17,868 original documents; by document-content hash a 98:1:1 ratio split yields train/validation/test of 17,440/167/178 books respectively. After further excluding books with BPE unknown-fragment ratio exceeding 1%, the three sets retain 17,332/167/176 books. Both models share the official GPT-1 BPE vocabulary of 40,478 entries. Each book is independently split into non-overlapping 512-token blocks, not concatenated across books, with the final short tail discarded; the first 16 tokens of each block serve as context and the latter 496 tokens for evaluation.

Baseline model. The comparison baseline, denoted the GPT baseline, is a model implemented strictly to the GPT-1 specification and scaled as needed for comparison: the 13M-level comparison is of the same scale as BriLLM-MetaPred (about 13.2M parameters), for controlled training-budget comparison; the 129M-level comparison follows the official GPT-1 parameter scale. The baseline

and BriLLM-MetaPred use the same data split and vocabulary, differing in model architecture and training objective. All "GPT baseline" below refers to this kind of comparison model.

H.2 Training–Inference Coupling

This section examines the engineering execution of training–inference coupling: the training checkpoint is held fixed, and only the evaluation-time allowance for fast-parameter update and state correction is switched. Table 7 gives a paired running diagnostic of a BookCorpus intermediate checkpoint.

Table 7: Running diagnostic of a BookCorpus intermediate checkpoint. Seed 42, step 100, 16 validation blocks, 7,936 evaluation tokens; memory is approximate peak allocated.

Metric	FROZEN	ADAPTIVE
NLL ↓	8.512188	8.512103
True evolution count	8,256	8,256
Simulation call count	3,120	3,120
Simulated measurement count	528	528
Direct measurement count	64	64
Fast-parameter update count	0	**1,056**
State correction count	0	**528**
Peak allocated memory (MB)	≈ 112	**≈ 175**

ADAPTIVE actually added 1,056 fast-parameter updates and 528 state corrections, and recorded higher memory occupancy; the checkpoint's direct-measurement residual average relative error is 0.653, above the 0.5 calibration threshold, and 64 corrections driven by untrustworthy direct measurements were suppressed. FROZEN also performs diagnostic measurement; the difference between the two columns' costs reflects only additional adaptation and correction, and cannot be used to estimate the entire measurement module's overhead.

H.3 Language Generation Examples

Chinese same-input generation comparison. Table 8 gives the outputs of 13M BriLLM-MetaPred and the corresponding GPT baseline. Both training seeds are 42, receiving the same complete prompt, greedily continuing 50 tokens, with prompts independently reset and without teacher forcing.

Table 8: Chinese same-input generation comparison.

Input prompt	BriLLM-MetaPred	GPT baseline
01 中国队在第二场比赛中3比0大胜	。此外，中国队还有1000万人次，中国队在2008年10月份以来，中国队在2008年10月份以来首次	日本队，中国队在第三场比赛中以1比0战胜日本队，中国队在第四场比赛中以1比0战胜日本队，中国队在第四
02 欧盟决定为经济疲弱的成员国提供总额达	到1.2亿美元。此外，美国政府将继续保持稳定增长。此外，美国政府将在2010年底前将有望进一步加强国	到1000亿欧元的纾困金，这是欧盟决定为经济增长提供的最大障碍。欧盟委员会主席巴罗佐表示，欧盟委员会
03 最近中医膏方养身保健成为了	保障房的安全保障，保障房建设保障性安居工程建设安全保障工程建设安全保障安全保障措施保障安全保障安全保	一个新的消费方式，但是，这种方式的确是一种新的消费方式，但是，这种方式的确是一种新的消费方式，但是，
04 消费者在购买家具的时候非常注重	要的原因是，这些都是在网上看到的。""在这个时候，我们看到了这个消息，我们看到了这个时候，我们看到了	家具的质量，而且家具的质量也是非常重要的。"在家具行业，家具的质量是否符合国家标准的标准，家具的质量

Table 8 (continued)

Input prompt	BriLLM-MetaPred	GPT baseline
05 儿童房甲醛超标5倍，即有引发	网络购物网络购物网站，网络购物网站、网络购物网站、网络网络建设、网络网络建设网络建设网络建设网络建设	甲醛的可能。”“甲醛超标”是一种新型甲醛释放量超标的新型甲醛释放量超标，这种新型甲醛释放量超标的新型
06 双色球一等奖的奖金有望达到	1000万美元。而在这个过程中，我们认为，这样的结构性机会是一个非常有效的解决方案。“我们的目标是在	1000万元。据悉，这是中国彩票双色球第10009期开奖的一张彩票，中奖号码为“0311111111

Both models can form local coherence of news style after some prompts, but both show repetition, topic drift, and unfounded numbers.

English same-input generation comparison. Table 9 gives the same-prompt continuations of two about 13M models on BookCorpus. Both take training seed 42 and the 788-step checkpoint; the prompt, under the same sampling seed, is the first 16 BPE tokens of a fixed validation block; each continuation is 128 tokens, with temperature 0.8 and top-k 40; prompts are independently reset. BriLLM-MetaPred uses the ADAPTIVE mode, and the generation process does not read subsequent reference tokens. The main text presents the first six in fixed order, each retaining the full continuation, with line breaks marked ”[/R]”.

Both models can produce English phrases and local coherence, and both show punctuation repetition, grammatical mismatch, and long-range semantic incoherence.

Table 9: English same-input generation comparison.

BriLLM-MetaPred	GPT baseline
01 Common input: gregory zorzos [/R] 100 alphanumeric crosswords [/R] gregory zor	
_ - one and the same one of your body than to see the the room - at a day of his feet ? ” [/R] ” , ” no , the house and the house of the side . and the day of the world . [/R] ” [/R] one of his life , we were in a different day , and the way you ’re sure , ” if the way had out for a long time to do , we do n’t think it . i have the door and did n’t be back at the world . but it was some important to be more . and this we ’m not a bit of the rest , and the room was in the small	- the , [/R] he said _ [/R] , as but it . [/R] _ ” yes , and the one . [/R] chapter 4 [/R] the ground , a good - 2 : [/R] the floor of the other the air . [/R] it , and the day that ’ [/R] ’ [/R] ’ : * * [/R] the last day , this ’s an year is the house . he is his first and he was not not like the most of the door and the same . if i have . [/R] ” i ’ll have no good _ , the air he was a bit of them from the most of the first , and i have to be going to get a lot
02 Common input: lover ’s knot . ” [/R] ” i ’m serious , morris . i want a	
second world , i do n’t know that it ’s been a new , ” _ _ _ _ he would see a good man and do what it is to be a moment . he was a few of the floor of the other , and how happened to you do n’t be . ” you know , she ’s not good . [/R] i need the last place i know who ” , ” i ’m a idea what she asked . ” [/R] ” what was there or this he was a few hundred than i have just to understand her . ” [/R] ” she said , ” if you have it . _ ” [/R] ” she is a better more than a	right . ” i do n’t see you , ” i do n’t have to it was too . . ” she would have to a a time to do you and i know ... you want to be a little . ” i ’m n’t know ... ” [/R] ” [/R] ” [/R] ” [/R] you have to know you . ” [/R] ” i ’re n’t go the same , her hair and i ’m n’t want to take a more in a good . you had be one than about you . ” why , ” she wanted to talk with a place . ” she was a little to him , he was still like a lot to go to i said . i
03 Common input: ibn sa’ûd assembled muslims and , slandering them , said ,	

Table 9 (continued)

BriLLM-MetaPred	GPT baseline
” i have come to the one . the long thing . [/R] ” i are a little part it is on to be an . the new . [/R] we ’re all the day . ” what he ’s not so , you ’re going to be going to give a time if you do . ” [/R] ” we ’ve been for what was to be . [/R] ” i did n’t know that , he had seen that all the ground . ” [/R] ” i had my own , ” ’s sure she ’d never been a good man . [/R] ” even that , to have not much to get on my father . ” [/R] * ” he said . ” [/R]	and the two a great , they are n’t keep , that is a most of the other , [/R] and for your way that was to the way to you have to the same , that ’s just a very more than you are you to be the way to have the table of the rest of the people is . [/R] but this and they were the house , and a the woman , the first of the little day , who in a whole . [/R] that had been the other ; . you were and the other of the table . [/R] no , when it , in the the next , he was a life , , the time , and her for
04 Common input: sured caressing . ” ah . i see , ” he said slowly , a trace	
of the man was n’t not here at the house . [/R] ” i go to the two years . i ’m already just a one of their side . she did n’t need to be in the two , i have done to the table and the man would n’t look . ” [/R] ” i ’d thought i could n’t make what i was to be able to see the two thing . the good day . [/R] they ’d found your eyes . ” the other face is a little the night and he told us for him . but ” [/R] ” we had to the one , the other . he looked into the air . ” he said . ” i ’m	, ” [/R] ” i ’d made to you did n’t know . ” [/R] ” [/R] ” he was not to stop you . ” [/R] ” [/R] ” she could n’t . [/R] ” i could n’t you you know she ’s all . ” [/R] ” he had her , ” [/R] ” [/R] ” let her , ” [/R] ” i did i ’m the first ? ” [/R] ” [/R] ” she ’d be . they was going to do n’t be like a good . ” i want to the ground . they do not in this . ” [/R] ” i ’m that they did n’t know . ” do you do n’t know i have to think you have to
05 Common input: - looking mahogany bar off to the side . ” let ’s get a drink .	
[/R] ” i mean . [/R] ” i do n’t know . ” [/R] ” you have to give me to do that . ’ [/R] i ’ll be on you . ” [/R] ” she said ; ” it , ” said that , not . ” what you do n’t do you . ” [/R] ” what will i have to be that for you at the way , he , ” ’s right . ” [/R] ” let it you be , ” you want to make a bit . ” she ’s all n’t i ’ll have to be it and you , just . ” [/R] ” you are you not a place and you . ” we can just like that ,	” i n’t believe it was you . ” [/R] ” i ’ll n’t get to get his life . ” it ? ” you ’re n’t want to her , ” [/R] ” [/R] ” [/R] when you to look . ” she ’s face . ” what was n’t do n’t have to be going to you were a few of the world ’s hand of the way to you could n’t go into the last a more i know he took her face with it is it ’s just just . she had been out of me , my head the first . ” [/R] ” she ’s n’t have you do anything and i could n’t like you know it will hear her ,

Considering context sensitivity, in the English case, BriLLM-MetaPred and the GPT baseline are on par in output performance. But in the Chinese case, the GPT baseline shows a stronger context-awareness (i.e., the model’s memory ability), while BriLLM-MetaPred is considerably weaker here. This can be understood as its capability leaning toward creative, free-form language generation. The character of the two models’ language-generation ability awaits further exploration in future work.

In sum, the experimental results support the paper’s goal on four fronts: (1) at matched training budget, the continuous-execution mode’s BriLLM-MetaPred reaches PPL comparable to the equal-size GPT baseline across two seeds; (2) in core ablation, disabling self-measurement markedly worsens FROZEN-mode prediction, supporting self-measurement as an effective endogenous dynamical check signal; (3) running diagnostics show that training–inference coupling indeed occurs, and fast-parameter update and state correction both run normally and are suppressed by calibration error; (4) at about 129M parameters the model completes training and evaluation and generates locally coherent text. These four results jointly support the three core features listed in Section 1, and constitute first evidence of the engineering feasibility of the first language-model realization of MetaPred.

I REPRODUCIBILITY CONVENTIONS

This section specifies key implementation details such as random-number generation, checkpoint saving, and evaluation conventions, to ensure the reproducibility of BriLLM-MetaPred experiments.

I.1 COMPLETE CONFIGURATION AND REPRODUCTION RECORD

The preceding sections give the main comparisons and their explanations. This section provides the training configuration and full precision in one place for verification.

I.1.1 Model Realization, Data, and Training Configuration

The experimental realization is based on the state network, controller network, and training algorithm of BriLLM-MetaPred; differences among experiments lie in training mode, parameter scale, training length, and evaluation mode, with concrete configuration in the tables of this section.

Table 10: Main experimental configurations. Parameter counts are registered parameters.

Experiment	Parameter count	Node count / width	Vocabulary	Actual training tokens
BookCorpus continuous-execution	13.2226M	96 / 132	40,478 BPE	25,014,272
THUCNews sparse-auxiliary 13M	13.0945M	256 / 256	6,512	25,000,960
THUCNews sparse-auxiliary 129M	129.3854M	256 / 928	6,512	470,404,384

BookCorpus data split and preprocessing. Of the 17,868 original documents of BookCorpusOpen, 17,785 remain after exact whole-book SHA256 deduplication; by document hash a 98:1:1 split yields train/validation/test of 17,332/167/176 books after preprocessing. The vocabulary is the official GPT-1 BPE of 40,478 entries; preprocessing uses spaCy 3.8.7 blank-en and ftfy 6.3.1.

Blocks lie within the same book, with length and stride both 512 and the short tail discarded; the 496 next-token targets at zero-based positions 16–511 are evaluated. The training set totals 2,903,718 blocks and 1,440,244,128 training tokens; the full validation set is 25,540 blocks and 12,667,840 tokens; the test-set metadata is 29,844 blocks and 14,802,624 tokens. Each step's global batch size 64 corresponds to 31,744 training tokens; 788 steps give an actual training token count of 25,014,272.

For the 13M comparison model, BriLLM-MetaPred's state network has node count/width 96/132, with 16 input and 16 output nodes, 40 modal nodes, and FFN width 528; the GPT baseline has 12 layers, width 192, 3 attention heads, FFN width 768, and dropout 0.1.

The shared optimizer is AdamW, peak learning rate 2.5×10^{-4}, with 40-step linear warm-up followed by cosine decay, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.01, and global gradient clipping 1. Training uses FP32 with TF32 allowed; formal evaluation disables TF32. BriLLM-MetaPred uses SDAA's 64×1 topology with flattened CPU FP32 Gloo mean synchronization; GPT uses A100's 2×32 CUDA DDP.

BriLLM-MetaPred has 5 autonomous-evolution steps, executing 1 generation evolution per training token; fast-parameter rank 8, local learning rate 0.05, clipping cap 1. History capacity 64, candidate-record pool 8, pending-prediction capacity 8; short-range/long-range simulation 2/5 steps, direct-measurement calibration interval 8, measurement/update budget 4096/256, coverage interval 32. Correction strengths are $\{0, 0.25, 0.5\}$, absolute RMS cap 0.05, relative natural-evolution cap 0.25.

Internal objective coefficients are $\lambda_{\text{pred}} = 1$, $\lambda_{\text{fp}} = 0.1$, $\lambda_{\text{dyn}} = 0.5$, outer intrinsic loss coefficients 0.01/0.1. State, measurement, and energy trend weights are 1/0.01/0.1; local state trend coefficient 0.001, direct-measurement calibration coefficient 1. The independent controller uses Adam, executing PPO every 64 steps, 2 epochs, mini-batch 32, clip 0.2, discount 0.99, GAE 0.95, value coefficient 0.5, entropy coefficient 0.01; training reward additionally includes 0.01 task cross-entropy feedback. Formal evaluation freezes the controller and stops PPO data collection.

THUCNews data split. THUCNews-v3 is split 90:5:5 by exact and conservative near-duplicate article groups, with train/validation/test of 728,267/40,382/40,533 articles; the vocabulary consists of the 6,512 tokens occurring at least 5 times in the training set. Full hyperparameters and optimizer configuration appear in Appendix I.1.

Execution details of sparse-auxiliary mode. The 13M sparse-auxiliary model's state network uses 256 nodes and node width 256; training on 25,000,960 tokens. The 129M model's state network is 256 nodes with node width 928.

I.2 Random Number Generation and Management

MetaPred involves two kinds of randomness: the inherent randomness of parameter initialization, and random operations during training/inference (such as policy sampling and dropout). To ensure reproducibility, the following management mechanisms are used:

- **Global RNG**: used for parameter initialization, controlled by the main seed (e.g., 42).
- **Runtime RNG**: each sequence runtime z_u contains an independent RNG state ξ_u. This state updates as the evolutionary step advances, ensuring:
  1. simulated evolution (self-measurement) and true evolution have completely isolated random streams;
  2. different running modes of the same sequence (TRAIN/FROZEN) produce identical random decisions (such as dropout masks) at the same evolutionary step, provided the RNG state is correctly saved and restored.
- **Policy sampling**: the controller's action sampling is based on the runtime RNG, ensuring causal consistency (i.e., sampling depends only on current and past information).

I.3 Checkpoint Format and Content

A training checkpoint must save the following state, to support resuming training from any evolutionary step or performing causally consistent evaluation:

- Slow parameters $\boldsymbol{\theta}$ (containing all learnable weights and biases).
- Control parameters $\boldsymbol{\psi}$.
- Optimizer state (AdamW first and second moment estimates).
- Current evolutionary-step counter u, token-step counter t, parameter-group index q.
- The complete state z_u of the most recent sequence runtime (including $\mathbf{S}_u, \boldsymbol{\phi}_u, \mathcal{H}_u, \mathcal{B}_u, \xi_u$), for resuming online evaluation.

The checkpoint filename contains the global step and a state label; alternative updates rejected by the slow-update preview mechanism are saved separately.

I.4 Evaluation Conventions

MetaPred's three running modes (TRAIN, FROZEN, ADAPTIVE) correspond to different evaluation purposes, with the following conventions:

- **TRAIN mode diagnostic**: computes task loss $\mathcal{L}_{\text{task}}$ and self-measurement loss $\mathcal{L}_{\text{self}}$ on the validation set, for monitoring training convergence. This diagnostic follows the full configuration of TRAIN mode (including fast-parameter update and state correction), keeping slow parameters fixed; it is a reading of the training-process state.
- **FROZEN mode evaluation**: freezes all parameters $(\boldsymbol{\theta}, \boldsymbol{\phi}_u, \boldsymbol{\psi})$, disables fast-parameter update and state correction, and executes dynamical evolution, self-measurement, and prediction read-out. External task loss may still be computed under this mode, but only for evaluating model capability; the self-measurement residual is a direct measure of the model's inner consistency at the current state. Controller actions are still computed and recorded, but produce no actual effect.
- **ADAPTIVE mode evaluation**: simulates the real deployment scenario. Freezes slow parameters $\boldsymbol{\theta}$ and control parameters $\boldsymbol{\psi}$, but allows fast-parameter update and state correction. During evaluation, the sequence runtime state z_u is reset at the sequence boundary. Used to measure the model's adaptive capability in online-learning scenarios.

I.5 Hardware and Numerical Precision

- **Hardware environment**: experiments are conducted on NVIDIA A100, A800, and SDAA, with CUDA 11.8 or above.

- **Numerical precision**: current formal experiments uniformly use `float32` (FP32) precision.
- **Deterministic algorithms**: in experiments with the highest reproducibility requirements, CuDNN deterministic mode is enabled (`CUDNN_DETERMINISTIC=1`), at some cost to training speed.

I.6 Code and Data Availability

The model realization is based on the PyTorch framework. Training corpora use the public datasets BookCorpusOpen, text8, and THUCNews-v3, following their respective data-use agreements. Preprocessing scripts, model definitions, and training/evaluation code will be made public with the paper, to ensure full reproducibility.