# Civilization Architecture: Structured Memory, State, and Rule Pathways for Auditable Hidden-State Computation

Yike Wang



***Abstract*—Large language model systems are commonly extended with retrieval, tool use, prompt templates, and parameter-efficient adapters, yet these mechanisms often leave memory, state control, and rule constraints outside an explicitly auditable latent computation path. This paper presents Civilization Architecture, a model-system architecture in which a Transformer-like neural core is coupled with structured Memory, State, and Rule pathways that enter hidden-state computation and expose path-specific traces for ablation and audit. The model is defined as the integrated computation graph, not as a standalone language model plus external orchestration. We evaluate the architecture through a staged validation protocol spanning a controlled PyTorch CivilizationTransformer and a frozen Qwen3-0.6B instantiation with trainable path-specific latent adapters. In the controlled backend, native Memory/State/Rule fusion supports hidden-state injection, chain-state alignment, and selective path-dependency stress tests. In the frozen-Qwen3 backend, the final Stage 42 multi-seed benchmark achieves raw full-hidden fixed-centroid recovery of 1.0000 accuracy over three seeds, with wrong-context drop 0.8000 and mean hidden-norm ratio 1.1270, while all Qwen3 base parameters remain frozen. Matched comparisons show that MLP adapters, LoRA, and prefix tuning reach the same accuracy ceiling, and context-token-only input reaches 0.9611 accuracy. Thus, the supported contribution is not predictive-performance superiority, but a native latent-path intervention and audit interface for structured memory and rule signals. Public benchmark transfer, longer-context robustness, lifelong graph-memory maintenance, and native full-architecture pretraining remain future work.**

***Impact Statement*—This work investigates a model-system architecture in which memory and rule signals are integrated inside the latent computation path rather than supplied only as external prompts, retrieval snippets, or tool calls. If validated at larger scale, this direction could support AI systems whose long-term state, rule constraints, conflict handling, and adaptation behavior are more auditable than standard prompt-level augmentation. The current evidence is intentionally bounded: experiments use a local Qwen3-0.6B neural core and controlled five-candidate benchmarks to test path-specific necessity, ablation behavior, and hidden-state integration. The manuscript therefore frames the contribution as a staged architecture and evaluation protocol for training and auditing civilization-like model systems, not as a completed general-purpose autonomous civilization system.**

***Index Terms*—Artificial intelligence, large language models, memory-augmented neural networks, neuro-symbolic systems, hidden-state auditing.**

Yike Wang is with Hydite AI Research, Haokir Inc., Colorado Springs, CO 80918, USA (e-mail: yikewang@research.hydite.com; ORCID: 0009-0009-1832-5421).

AI tools were used for drafting assistance, language refinement, LaTeX organization, and consistency checking; all technical claims, experiments, results, and final responsibility remain with the author.

## I. Introduction

Contemporary large language model systems often rely on external retrieval, agent orchestration, and prompt-level control to provide memory and behavioral constraints. These methods are useful, but they do not require the model to represent memory, state, or rules as distinct computational pathways. As a result, it is difficult to determine whether a decision used a memory signal, a state-control signal, a rule signal, or only a surface correlation in the input.

This paper investigates Civilization Architecture as a new model-system architecture. In this architecture, a Transformer-like language model is the neural computation core, but it is not the complete model. The full model is the coupled computation graph formed by the neural core, structured Memory, State, and Rule pathways, path-specific traces, readout, and audit mechanisms. Training such a system therefore means training the architecture-level computation paths, not only optimizing the weights of a standalone LLM. The project is not only a final adapter experiment. It includes a minimal reference backend, a trainable PyTorch CivilizationTransformer with native Memory/State/Rule fusion, and a frozen Qwen3-0.6B real-model instantiation using path-specific latent adapters. The Qwen3 instantiation is treated as a conservative stage in this architecture roadmap: the base decoder is frozen for auditability, while structured paths are still trained and evaluated as parts of the same model-system forward process.

The paper makes five contributions:

- We define Civilization Architecture as a model-system architecture composed of a Transformer-like neural core, Memory pathway, State pathway, Rule pathway, trace/audit surface, and readout protocol.
- We implement a controlled PyTorch CivilizationTransformer that replaces ordinary Transformer blocks with CivilizationBlock-style fusion and verifies Memory/State/Rule path dependency under hard logic and stress-test settings.
- We instantiate the architecture on frozen Qwen3-0.6B through path-specific latent adapters, preserving base weights while injecting structured signals into hidden-state computation.
- We introduce a fail-fast evaluation protocol with path ablations, wrong-context tests, surface/group flips, projected delta readout, and raw full-hidden fixed-centroid readout.

- We conduct parameter- and data-matched comparisons against an MLP adapter, LoRA, prefix tuning, context-token-only input, and task-text-only input, explicitly separating path-level controllability from predictive accuracy superiority.

The empirical boundary is deliberately explicit. The current results support controlled path necessity and hidden-state integration for this model-system architecture. The matched baselines do not show predictive superiority over standard adapters or parameter-efficient tuning. The results also do not claim public benchmark migration, completed native pretraining of a full Civilization foundation model, a completed lifelong learning system, or production graph-memory governance.

## II. Related Work

Civilization Architecture builds on Transformer language models [1] and the observation that modern decoders expose rich intermediate representations that can be inspected or modified. The present work differs from standard language-model adaptation in its evaluation target: the goal is not only to improve final output accuracy, but to test whether Memory, State, and Rule signals become necessary latent paths inside hidden-state computation. This makes the closest comparison methodological rather than purely performance-based: the question is whether a system exposes separable intervention surfaces for structured information sources, and whether those surfaces survive disabled-adapter, zero-scale, wrong-context, and target-path ablation controls.

### A. Retrieval, Tools, and Agentic Context

Retrieval-augmented generation (RAG) augments a language model with retrieved documents or passages [2], and retrieval-scaled language models such as RETRO demonstrate that large nonparametric stores can improve knowledge-intensive modeling [3]. Tool-using and agentic systems extend this idea by allowing a model to call external tools or interleave reasoning and acting steps [4], [5]. These systems are important external-context mechanisms, but the retrieved facts, tool outputs, and controller states are usually serialized into the input stream or managed outside the decoder. They therefore do not by themselves show that a decision depended on a distinct Memory, State, or Rule path. In contrast, the present work evaluates whether disabling a structured latent path changes the decision while irrelevant-path ablations remain controlled. The context-token-only baseline in this paper intentionally isolates one part of this distinction: the same grounded Memory/Rule/State content is serialized as ordinary input tokens without trainable parameters. Its near-ceiling accuracy shows that token context can be highly informative in the reported benchmark, but it does not provide a native latent intervention surface.

### B. Memory-Augmented Neural Models

Earlier memory-augmented neural models introduced explicit memory modules and differentiable read/write mechanisms [6], [7]. End-to-end memory networks further showed that neural models can attend over stored facts for question answering [8]. Civilization Architecture shares the motivation that persistent information should not be reduced to ordinary sequence tokens. However, the current paper focuses on Transformer hidden-state integration and path necessity rather than general memory read/write capacity. Memory is treated as one structured pathway among Memory, State, and Rule, and its contribution is tested through wrong-context, no-memory, adapter-disabled, and zero-scale controls.

### C. Parameter-Efficient Adaptation

Adapters [9], low-rank adaptation (LoRA) [10], prefix tuning [11], prompt tuning [12], P-Tuning v2 [13], and compact adapter variants [14] reduce the number of trainable parameters needed to adapt a frozen or partially frozen model. The frozen Qwen3 instantiation in this paper is related to this family because it trains adapter/readout parameters while keeping the base decoder unchanged. The distinction is that the adapter is decomposed into path-specific residuals $(\Delta_M, \Delta_S, \Delta_R)$ and evaluated as a causal test surface. A single adapter residual can improve accuracy while hiding which source supplied the decision signal; the reported protocol requires path-specific traces and targeted ablations. The matched-baseline experiments reflect this distinction. A generic MLP adapter, LoRA, and prefix tuning are treated as strong adaptation baselines rather than weak controls, and all reach the same accuracy ceiling in the current benchmark. The claimed difference is therefore not that Civilization Architecture is more accurate under this sample contract, but that its native Memory and Rule paths are separately addressable and auditable.

### D. Rules, Neuro-Symbolic Methods, and Governance

Neuro-symbolic models combine neural perception or representation learning with symbolic structure [15]. Rule and policy mechanisms also appear in alignment-oriented systems, including constitutional-style rule specifications [16]. These approaches motivate the need for rule-conditioned behavior, but many implementations apply rules as prompts, external policies, or post-hoc filters. The Rule pathway in Civilization Architecture is instead a model-internal input to the hidden-state update. This allows conflict tasks to test whether disabling Rule, rather than Memory or State, selectively damages the decision.

### E. Hidden-State Analysis and Steering

Mechanistic interpretability studies show that internal circuits and latent features can sometimes be localized and analyzed [17], while latent-knowledge and representation-engineering work studies whether model states contain directions that can be read or steered [18], [19]. The present work uses hidden-state readout and intervention tools, but it does not claim a universal semantic direction. Its narrower claim is operational: structured paths should produce measurable deltas, those deltas should support the intended readout, and the effect should fail under disabled, zero-scale, wrong-context, or target-path ablation controls.

## III. Problem Formulation

### A. Architecture Boundary

Civilization Architecture is defined as a model-system architecture in which a Transformer-like neural core is coupled with structured Memory, State, and Rule pathways. The neural core remains responsible for token-level representation, contextual hidden states, and language generation, but it is not treated as the entire model. The model instance is the coupled computation graph that includes the neural core, structured pathways, trace surfaces, readout modules, and, in the broader roadmap, memory evolution and review mechanisms. Let $x$ denote an input sequence and let

$$\mathcal{C} = (M, S, R)$$

denote structured context, where $M$ is a set of memory records, $S$ is a state-control vector, and $R$ is a set of rule records. A Civilization model computes

$$y = F(x, \mathcal{C})$$

through hidden states that are conditioned by these structured pathways rather than by the input sequence alone.

The project separates the architecture into three empirical levels. The first level is a minimal reference backend that implements a Transformer-like core, Memory items, State controls, Rule items, and a CivilizationBlock. The second level is a trainable PyTorch CivilizationTransformer in which blocks directly fuse Memory attention, State gates, and Rule projections. The third level is a frozen Qwen3-0.6B instantiation in which the base model remains unchanged while trainable path-specific adapters inject structured latent deltas into selected hidden layers.

### B. Training Object

The training object in Civilization Architecture is the whole architectural forward path, not only the parameter tensor of an isolated LLM. In the native controlled backend, training updates the CivilizationTransformer, including the CivilizationBlock fusion paths that consume Memory, State, and Rule signals. In the frozen-Qwen3 instantiation, the pretrained decoder is deliberately frozen so that the experiment can audit whether structured pathways change hidden states without changing Qwen3 itself. This does not make Qwen3 an external component: within the instantiated model, Qwen3 supplies the neural core and the path-specific adapters supply trainable architecture-level pathways coupled to its hidden stream.

This distinction is important for future scaling. A native Civilization model could train the neural core, Memory encoders, State gates, Rule paths, readout heads, and memory-evolution interfaces jointly or in staged curricula. The present paper validates the part of that training problem that can be isolated most cleanly: whether structured pathways can be made necessary, ablatable, and recoverable in hidden-state computation before claiming full native pretraining.

### C. Distinction From Prompt, RAG, and Agent Systems

Prompt engineering, retrieval-augmented generation, and agent orchestration can provide useful external context, but they usually leave the pretrained language model's latent computation unchanged. In such systems, retrieved facts, behavioral instructions, and tool outputs are serialized as additional text and then consumed by the same monolithic forward path. The model is not required to represent whether a decision came from persistent memory, temporary context, state control, or a rule constraint.

Civilization Architecture targets a different boundary. Memory, State, and Rule are represented as structured vectors, gates, attention inputs, residual updates, or readout traces. They are therefore part of the model computation interface, not merely part of the prompt. This distinction is central because the proposed evaluation asks whether a specific path is necessary for a decision, not merely whether additional text improves accuracy.

### D. Why Structured Pathways Must Enter Hidden-State Computation

Three requirements motivate placing Memory, State, and Rule inside the hidden-state path.

First, memory must be distinguishable from ordinary surface context. A model that receives memory only as text can use it as a correlated token pattern, and success does not show that the system has a memory pathway. By encoding memory as vectors or deltas and then disabling the Memory path, the experiment can test whether the target decision actually depends on that path.

Second, state must be measurable rather than described as a natural-language persona. Parameters such as rigor, creativity, defensiveness, rule priority, or write permission should affect gates, scales, attention, or readout behavior in a reproducible way. Otherwise, state control is indistinguishable from another instruction prompt.

Third, rules must influence the computation before the final output audit. Post-hoc filtering can reject an output, but it cannot show that a rule participated in resolving a conflict inside the representation. A Rule pathway allows rule-conditioned conflict tasks to test whether disabling Rule changes the decision while disabling Memory or State does not.

### E. Path Necessity

Let $p \in \{m, s, r\}$ denote the Memory, State, or Rule path. For a task group $g$, let $a_g(\text{full})$ be full-context accuracy and let $a_g(\neg p)$ be accuracy when path $p$ is disabled. The path drop is

$$D_{g,p} = a_g(\text{full}) - a_g(\neg p).$$

A path is considered necessary for group $g$ only when

$$a_g(\text{full}) \geq \tau_a, \qquad D_{g,p} \geq \tau_d, \qquad D_{g,q} \leq \epsilon \tag{1}$$

for irrelevant paths $q$. The group must also satisfy wrong-context or counterfactual controls when the task is designed to flip under changed structured context. This definition prevents

a high aggregate score from being explained by surface leakage, path substitution, or a readout that rebuilds its own codebook under each ablation.

### F. Current Empirical Boundary

The current evidence supports four claims. First, Memory, State, and Rule can be encoded, executed, trained, traced, and ablated in controlled Transformer-like models. Second, hidden-state representations can be observed, clustered, and intervened on under leakage controls. Third, the PyTorch CivilizationTransformer shows native Memory/State/Rule path dependency under controlled stress tests. Fourth, the frozen Qwen3 instantiation shows that structured Memory and Rule signals can be injected into a real language model's hidden-state stream without changing base weights, and can pass controlled five-candidate group necessity gates.

The formulation does not claim that a full native Civilization foundation model has already been pretrained, that Qwen3 base weights have been updated in the frozen instantiation, that public benchmark transfer is solved, or that graph memory maintenance, lifelong learning, audit isolation, and human approval loops are complete empirical results. These components are part of the broader architecture boundary and future work, while the present paper focuses on structured latent pathways, path necessity, and controlled hidden-state integration.

## IV. Civilization Architecture

This section specifies Civilization Architecture as a model-system architecture rather than a prompting pattern or an external orchestration layer. The central design choice is that Memory, State, and Rule are represented as structured computational paths that interact with hidden states. Textual prompts, retrieved passages, and tool outputs may be used as sources of these paths, but they are not the paths themselves. The neural core, structured pathways, and readout/audit interfaces are therefore designed as one trainable architecture, even when the current real-model instantiation freezes the neural core for controlled validation.

### A. Overall Civilization Architecture

Fig. 1 gives the overall architecture. An input sequence is encoded by a neural core, while three structured pathways provide additional model-internal signals. The Memory pathway supplies long-range factual, episodic, or graph-derived context. The State pathway supplies continuous control variables, such as reasoning mode, conflict pressure, or policy strength. The Rule pathway supplies constraints, preferences, and conflict-resolution structure. The output layer is deliberately separated from the audit layer: the former produces task outputs, whereas the latter measures which path changed which hidden-state region and whether the expected counterfactual controls hold.

The architecture has two implemented instantiations in the current project. The controlled instantiation replaces standard Transformer blocks with CivilizationBlocks and is used for direct ablation and stress testing. The frozen-LLM instantiation attaches path-specific adapters to selected layers of a frozen Qwen3 model and is used to test whether structured signals can be injected into a real decoder hidden stream without updating the base model. In both cases, the unit under study is the integrated forward graph: the Qwen3 backend is not a separate LLM plus a post-hoc wrapper, but a constrained instantiation in which the frozen decoder provides the neural core and the trainable pathways implement the Civilization-specific structure.

### B. Neural Core

The neural core is a Transformer-like sequence model that provides token embedding, self-attention, MLP transformation, residual computation, and output readout. In the controlled backend, the neural core is a compact Transformer whose ordinary blocks can be replaced by CivilizationBlocks. In the Qwen3 backend, the neural core is the pretrained Qwen3 decoder, including its tokenizer, embedding layer, decoder blocks, and language-model head.

Civilization Architecture does not require the neural core to store all persistent knowledge or policy structure inside its parameters. Instead, the neural core acts as the differentiable substrate through which structured context can be expressed. The architectural interface is the hidden-state stream $H_0, \ldots, H_L$, not the surface prompt alone. This is why the project tests hidden-state codebooks, direct hidden injection, full-hidden centroid readouts, and path-specific residual deltas rather than relying only on generated text.

### C. Memory Pathway

The Memory pathway maps external or persistent records into vectors that can be attended to by the neural core. In the controlled implementation, memory items are encoded as dense vectors and consumed through a memory-attention branch inside each CivilizationBlock. In the Qwen3 implementation, memory strings are encoded through a frozen context encoder and supplied to the adapter as separate memory vectors and masks. The memory path can then produce a path-specific residual update and a memory-attention trace.

This design differs from retrieval-augmented generation. RAG typically inserts retrieved text into the prompt context and lets ordinary self-attention process the enlarged sequence. Civilization Architecture treats retrieval as only one possible source of Memory. The Memory pathway is defined by its computational role: it must produce model-internal representations that can be isolated, ablated, counterfactually swapped, and traced in hidden-state space.

### D. State Pathway

The State pathway represents continuous control variables that modulate computation. In the controlled backend, state variables are transformed through state gates and projections that scale or add state-dependent context to the block output. In the Qwen3 adapter, state values drive learned gates over base, memory, rule, and state residual paths. This makes State a computational condition rather than a natural-language instruction such as "be careful" or "follow rule X."

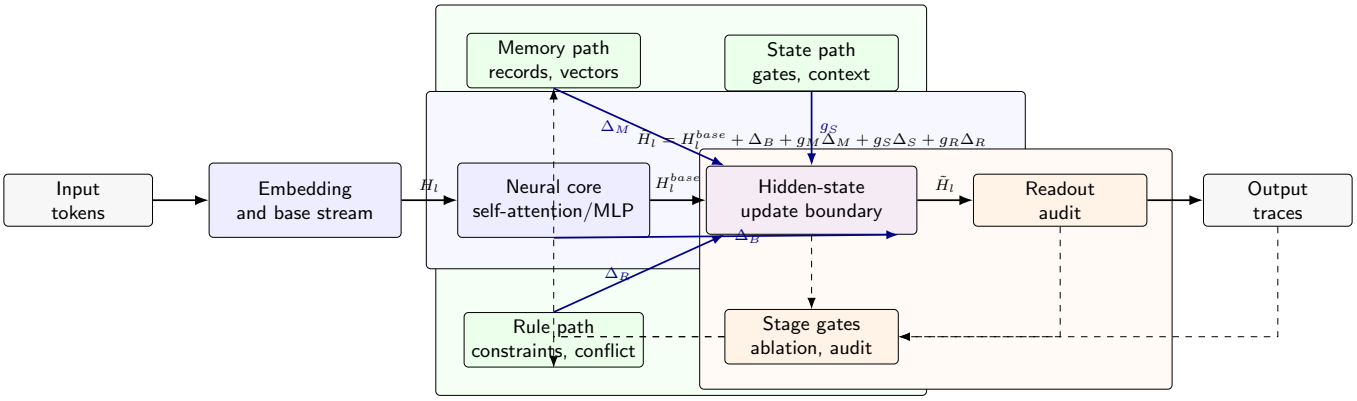


Fig. 1. Overall Civilization Architecture. Memory, State, and Rule are structured pathways that enter hidden-state computation, while readout and audit modules evaluate the effect of each path. Dashed components denote long-term learning, staging, and memory consolidation mechanisms that remain outside the current empirical claim.

State is necessary because many failures observed in the experiments are not purely factual failures. Some require mode control: whether to follow memory evidence, whether to privilege a rule under conflict, whether to resist surface template leakage, and whether to maintain chain-state across multi-step examples. A prompt-only representation of these modes is difficult to audit because the same token string can be interpreted differently across contexts. The State pathway gives the architecture an explicit variable whose influence can be disabled, zero-scaled, or measured.

### *E. Rule Pathway*

The Rule pathway encodes constraints and conflict-resolution structure. In the controlled backend, rule vectors are projected into the block computation and their influence is recorded in the trace. In the Qwen3 backend, rule vectors enter a separate cross-attention path and produce a rule-specific residual delta. The Rule pathway is tested both as an information source and as a necessary path under conflict conditions, where the correct answer should change when rules are changed but should remain stable when irrelevant paths are ablated.

Rules are not treated as final symbolic overrides. They are structured inputs to neural computation. This keeps the architecture compatible with gradient-based training and hidden-state analysis, while still allowing the paper to ask a symbolic question: whether a rule path is causally necessary for the observed decision.

### *F. CivilizationBlock*

Fig. 2 shows the controlled block. Let $H_l$ be the incoming hidden states, $M$ the memory vectors, $S$ the state vector, and $R$ the rule vectors. A CivilizationBlock first applies the ordinary Transformer update $T_l(H_l)$ and then adds structured path contributions:

$$H_{l+1} = T_l(H_l) + \Delta_M(T_l(H_l), M, S) + \Delta_S(S) + \Delta_R(R, S). \quad (2)$$

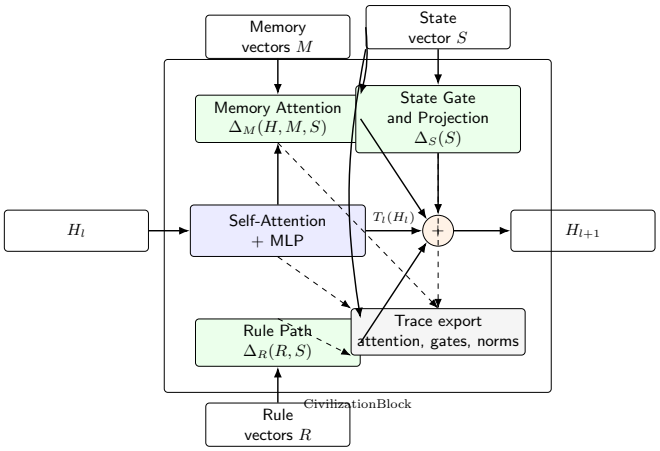


Fig. 2. CivilizationBlock in the controlled backend. A standard Transformer update is augmented by Memory attention, State gating/projection, and Rule projection before residual output and trace export.

The exact implementation can vary by backend, but the required contract is that each structured path has a separately controllable interface and a traceable effect on the hidden state.

The current PyTorch implementation records self-attention, memory attention, state scales, and rule-influence norms. Ablation flags can disable Memory, State, Rule, chain-state objectives, priority-control objectives, centroid separation, and hard-negative training. The block is therefore not only a forward-computation module but also an experimental unit for path dependency.

### *G. Hidden-State Injection*

Hidden-state injection is the mechanism by which structured signals are tested inside the neural computation rather than only at the prompt boundary. Earlier controlled stages construct hidden-state codebooks, test whether option directions can be recovered, inject latent vectors into intermediate states, and verify that repaired training objectives remove template leakage. Later Qwen3 stages move this idea to a frozen decoder by using forward hooks and adapter residuals at selected layers.

The paper uses hidden-state injection conservatively. It does not claim that a single latent direction universally represents a concept across all tasks. Instead, it treats hidden-state interventions as testable interfaces: a path should produce a measurable delta, the delta should support the target readout, and the effect should disappear or change under disabled-adapter, zero-scale, wrong-context, and target-path ablation controls.

### H. Path-Specific Adapter for Frozen LLMs

Fig. 3 shows the frozen Qwen3 instantiation. The base model remains frozen. Structured context is encoded outside the base decoder and injected through adapters attached to selected hidden-state layers. The strongest adapter variant separates four residual paths:

$$\tilde{H}_l = H_l + \Delta_B(H_l) + \Delta_M(H_l, M, S) + \Delta_R(H_l, R, S) + \Delta_S(S). \tag{3}$$

Each path has its own learned residual scale, and the adapter exports base-, memory-, rule-, and state-delta tensors for downstream audit. The formalized frozen-core and path-fusion equations are given in Section V.

The path-specific design is important because a single adapter residual can hide whether an observed improvement came from Memory, Rule, State, or an unstructured base correction. Separating the residual paths turns the adapter into a causal test surface. The implementation also checks adapter-disabled equivalence, zero-scale behavior, hidden-norm limits, hook removal, and path-specific dominance so that a successful stage cannot be explained by hook leakage or uncontrolled hidden-state drift.

### I. Readout and Audit Protocol

The architecture includes readout and audit as first-class methodology rather than post-hoc visualization. Two readout families are used. Projected delta readout tests whether a path-specific residual delta carries the expected option direction. Full-hidden fixed-centroid readout tests whether the signal remains recoverable from the final hidden representation using train-only centroids and test-condition reuse of those fixed centroids. These readouts answer different questions: the first validates the injected path signal; the second tests whether that signal survives integration with the frozen model's hidden stream.

Each stage is paired with negative controls. The project uses disabled adapters, zero residual scale, no-memory, no-rule, no-state, irrelevant-path ablation, wrong-context swaps, counterfactual groups, and group-level fail-fast gates. Failures are retained as part of the empirical record, including hard benchmark failures, local real-task failures, and multiclass integration failures. This audit protocol defines the current empirical boundary of the method: the paper claims verified path-specific hidden-state integration under these controlled conditions, while long-term autonomous learning, graph-memory consolidation, deployment-time staging, and human-governed memory evolution remain future extensions.

## V. Mathematical Formulation

### A. Structured Context and Frozen Core

Let $x = (x_1, \ldots, x_T)$ be an input sequence. A Transformer-like neural core with parameters $\theta$ computes hidden states

$$H_0, \ldots, H_L = f_\theta(x), \qquad H_\ell \in \mathbb{R}^{T \times d}. \tag{4}$$

In the frozen-Qwen3 experiments, $\theta$ is held fixed:

$$\nabla_\theta \mathcal{L} = 0, \qquad \theta^{(t+1)} = \theta^{(t)}. \tag{5}$$

This condition is enforced in the implementation by excluding Qwen parameters from optimizers, recording zero Qwen gradients, and rejecting checkpoints that contain base-model weights.

Structured context is represented as

$$\mathcal{C} = (M, S, R), \tag{6}$$

where $M = \{m_i\}_{i=1}^{n_m}$ are Memory vectors, $S \in \mathbb{R}^{d_s}$ is a State vector, and $R = \{r_j\}_{j=1}^{n_r}$ are Rule vectors. In the current Qwen3 implementation, these vectors are derived by a frozen context encoder from structured local task records; the mathematical interface does not require that the source be text.

### B. CivilizationBlock Update

In the controlled backend, a CivilizationBlock augments an ordinary Transformer block $B_\ell$ with structured paths:

$$H_{\ell+1} = B_\ell(H_\ell) + \Delta h_M + \Delta h_S + \Delta h_R, \tag{7}$$

where

$$\Delta h_M = A_M(B_\ell(H_\ell), M, S), \tag{8}$$

$$\Delta h_S = G_S(S), \tag{9}$$

$$\Delta h_R = P_R(R, S). \tag{10}$$

$A_M$ denotes Memory attention, $G_S$ a state projection or gate, and $P_R$ a state-conditioned Rule projection. The required architectural contract is not a particular parameterization of these functions, but that each path has a separately controllable contribution to hidden-state computation.

### C. Path-Specific Frozen-Model Instantiation

For selected frozen-model layers $\ell \in \mathcal{L}$, the adapter observes $H_\ell = f_{\theta,\ell}(x)$ and writes a decomposed residual update:

$$\tilde{H}_\ell = H_\ell + \alpha_B g_B(S) \bar{\Delta} h_B + \alpha_M \bar{\Delta} h_M + \alpha_S \bar{\Delta} h_S + \alpha_R \bar{\Delta} h_R, \tag{11}$$

$$\bar{\Delta} h_B = U_B(H_\ell), \tag{12}$$

$$\bar{\Delta} h_M = U_M(A_M(Q_M(H_\ell), M)), \tag{13}$$

$$\bar{\Delta} h_S = U_S(S), \tag{14}$$

$$\bar{\Delta} h_R = U_R(A_R(Q_R(H_\ell), R)). \tag{15}$$

Here $\bar{\Delta} h_B$ is an unstructured base-adapter correction, while $\bar{\Delta} h_M$, $\bar{\Delta} h_S$, and $\bar{\Delta} h_R$ are unscaled Memory, State, and Rule path updates. The learned residual scales $\alpha_B, \alpha_M, \alpha_S, \alpha_R$ and state-derived gates correspond to the adapter trace fields used for audit. In the Stage 40–42 frozen-Qwen3 evidence,

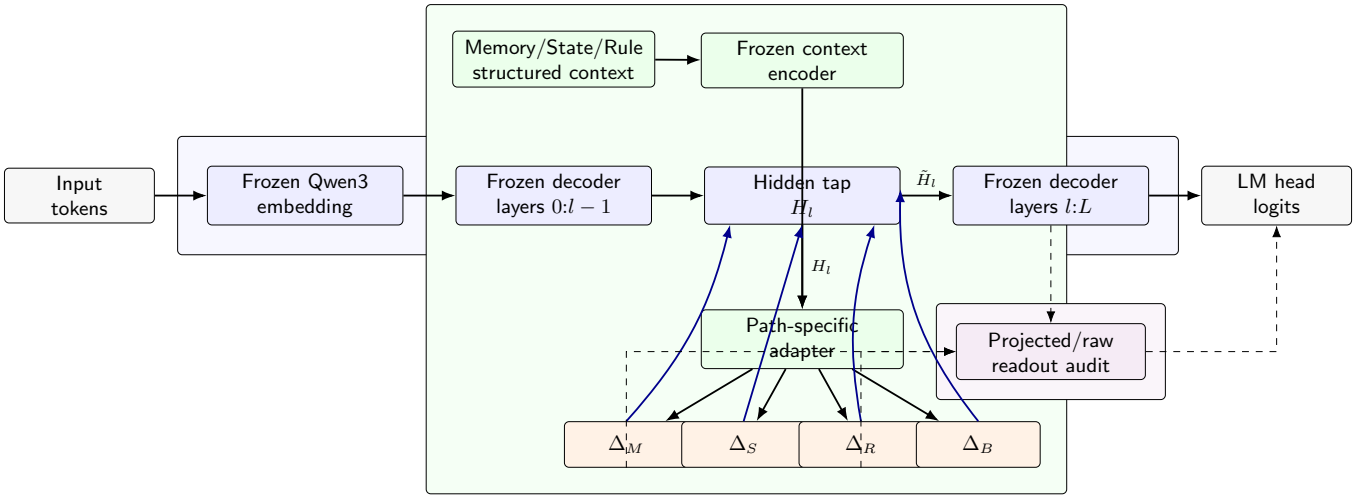


Fig. 3. Frozen Qwen3 instantiation. Qwen3 parameters remain frozen while selected decoder layers receive path-specific adapter deltas. Readout modules evaluate projected path deltas and full hidden states.

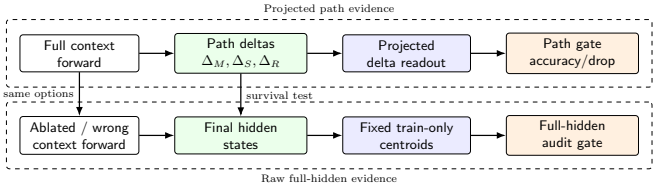


Fig. 4. Readout and audit protocol. Projected path readout validates the path-specific residual signal, while fixed train-only centroids test whether the signal survives in raw full hidden states under ablation and wrong-context controls.

Memory and Rule are the primary necessity-tested paths; State is retained as a gating/control interface and is also exposed to ablation, but it is not claimed as an independently validated necessity result at the same level.

A compact form used for discussion is

$$\tilde{h} = h + \alpha_M \bar{\Delta} h_M + \alpha_S \bar{\Delta} h_S + \alpha_R \bar{\Delta} h_R + \alpha_B \bar{\Delta} h_B, \quad (16)$$

with $\alpha_p = 0$ or $\bar{\Delta} h_p = 0$ under the corresponding zero-scale or path-ablation control.

### D. Path Necessity Gate

Let $g$ denote an evaluation group, $p \in \{M, S, R\}$ a path, and $q$ an irrelevant path for that group. Let

$$a_g(c) = \frac{1}{|E_g|} \sum_{(x_i, y_i) \in E_g} \mathbf{1}\{\hat{y}_i(c) = y_i\} \quad (17)$$

be accuracy under condition $c$, where $c = \text{full}$ uses all structured paths and $c = \neg p$ disables path $p$. The target-path drop is

$$D_{g,p} = a_g(\text{full}) - a_g(\neg p). \quad (18)$$

A path is accepted as necessary for group $g$ only when

$$a_g(\text{full}) \geq \tau_a, \quad (19)$$

$$D_{g,p} \geq \tau_d, \quad (20)$$

$$D_{g,q} \leq \epsilon_q, \quad (21)$$

$$D_{g,\text{wrong}} = a_g(\text{full}) - a_g(\text{wrong}) \geq \tau_w. \quad (22)$$

The final local five-candidate gates use this logic with target-path and wrong-context drops near $0.80$, while irrelevant-path drops are expected to remain small for the corresponding group. Adapter-disabled and zero-scale conditions provide additional implementation controls; they test whether the observed behavior disappears when the adapter surface is present but unable to write a residual.

### E. Projected Delta Readout

Projected delta readout tests whether an individual path residual carries the expected option direction before relying on the full hidden state. Let $\Delta h_{p,\ell,t}$ be the trace-exported residual for path $p$ at layer $\ell$ and token position $t$. The readout vector is

$$v_p = \text{pool}_{\ell \in \mathcal{L}, t} \left( \Delta h_{p,\ell,t} \right). \quad (23)$$

With learned projector $W_\phi$ and candidate option vectors $o_1, \ldots, o_K$, scores are

$$z_k = \langle W_\phi v_p, o_k \rangle, \qquad \hat{y} = \arg\max_k z_k. \quad (24)$$

This readout corresponds to the Stage 40 Memory-delta gate and the Stage 41 Rule/Conflict projected gates.

### F. Fixed-Centroid Full-Hidden Readout

Fixed-centroid readout tests whether the validated path signal survives inside the raw full hidden state. For each group type $g$ and candidate option $k$, centroids are built only from train split, full-context hidden states:

$$\mu_{g,k} = \frac{1}{|\mathcal{T}_{g,k}|} \sum_{(x_i, y_i) \in \mathcal{T}_{g,k}} \text{pool}\Big( \tilde{H}_L(x_i, \text{full}) \Big). \quad (25)$$

For a held-out or controlled evaluation example, the prediction is

$$\hat{y} = \arg\max_k \left\langle \frac{u}{\|u\|_2}, \frac{\mu_{g,k}}{\|\mu_{g,k}\|_2} \right\rangle, \quad u = \mathrm{pool}\Big(\tilde{H}_L(x,c)\Big). \tag{26}$$

The same centroid set $\{\mu_{g,k}\}_{k=1}^{K}$ is reused for full, held-out, wrong-context, counterfactual, adapter-disabled, zero-scale, and path-ablation conditions. No centroid is rebuilt from ablated or test-condition hidden states. This train-only centroid construction is the leakage-control rule used by the Stage 42 full-hidden integration and the Stage 42 multi-seed robustness benchmark.

## VI. Stage-Gated Audit Procedure

The validation procedure is fail-fast. A stage begins from the strongest previously accepted artifact, enables only the parameter subset allowed by that stage, and evaluates both research gates and engineering gates before any later claim is considered. The controlled backend first verifies executable Memory, State, Rule, and CivilizationBlock paths; finite hidden states; trace export; ablation switches; codebook generalization after leakage repair; hidden-state injection; MSR fusion; chain-state alignment; and path-dependency stress. A failed template, masked-keyword, split, path-dependency, or stress gate writes failure artifacts and prevents the corresponding hidden-state claim.

The frozen-Qwen3 route then records the local model fingerprint, freezes all base parameters, excludes Qwen3 from the optimizer, and attaches only the allowed adapter/readout modules. For each group, the audit runs full-context, target-path ablation, irrelevant-path ablation, wrong-context, adapter-disabled, and zero-scale modes. It scores projected path deltas when the stage asks whether a path residual carries the intended option direction, and scores raw full hidden states with fixed train-only centroids when the stage asks whether that signal survives integration into the hidden stream. A path is accepted only when full accuracy and group success pass, target-path drop is large, irrelevant drops are controlled, disabled/zero-scale controls collapse, Qwen3 weights remain unchanged, and Qwen3 gradients remain zero. The supplementary material gives the stage ledger, fail-fast gates, runner paths, and artifact manifest.

## VII. Experimental Setup

### A. Staged Validation Protocol

The evaluation is organized as a staged validation protocol rather than a single terminal benchmark. Each stage has an explicit entry condition, a backend-specific implementation boundary, a set of gates, and a recorded decision about whether later stages may proceed. This protocol is important for two reasons. First, it prevents a successful late-stage run from hiding earlier failures such as template leakage, insufficient path ablation, or readout mismatch. Second, it separates engineering validity from research validity: a stage can pass unit tests and artifact generation while still failing the research gate.

Table I summarizes the experimental route. Stages 01–03 establish minimal feasibility in reference and PyTorch backends.

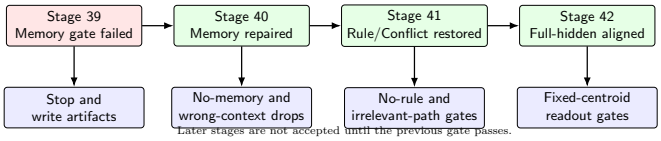


Fig. 5. Fail-fast repair protocol for the final Qwen3 stages. Later stages are not accepted until the previous path-specific gate has passed and failure artifacts have been preserved.

Stages 04–07 show that hidden-state codebooks are observable but also expose template leakage, which must be repaired before injection experiments are allowed. Stages 08–17 then validate hidden-state intervention, Memory/State/Rule fusion, CivilizationBlock replacement, hard logic failure discovery, chain-state repair, path-dependency ablation, and migration-precondition stress tests. Stages 18–26 move the method to a frozen Qwen3-0.6B backend and record mixed results: the adapter path can be trained under frozen-base constraints, but multilayer and real-task transfer gates expose failures. Stages 27–38 diagnose the failure source through path-specific adapters, projected readouts, fixed-centroid alignment, and five-candidate group gates. Stages 39–42 form the final fail-fast repair chain: Memory fails under strict gates, Memory is repaired, Rule/Conflict are restored, and raw full-hidden fixed-centroid integration is validated. A follow-up Stage 42 multi-seed robustness benchmark repeats the raw full-hidden integration gate over seeds 202, 303, and 404 without changing the base Qwen3 weights. Two matched-baseline rounds then compare the frozen-Qwen3 instantiation with a generic MLP adapter, LoRA, prefix tuning, context-token-only input, and task-text-only input under the same sample contract.

### B. Experiment Families

The evaluation follows the staged structure of the project rather than a single terminal experiment.

- The reference backend verifies that Memory, State, Rule, and CivilizationBlock components can be encoded and executed.
- The PyTorch CivilizationTransformer backend evaluates hidden-state codebooks, hidden-state injection, Memory/State/Rule fusion, chain-state alignment, hard logic benchmarks, path dependency, and stress profiles.
- The Qwen3 backend evaluates whether the same structured pathways can be instantiated inside a frozen real language model through adapters and readouts.

### C. Models

The controlled backend uses trainable Transformer-like models with CivilizationBlock fusion. The real-model experiments use a local Qwen3-0.6B checkpoint [20]. In the Qwen3 setting, the base model, tokenizer, and embeddings remain frozen; only adapter and readout parameters are trained. The full seed, hardware, runtime, runner, and artifact ledger is reported in the supplementary material.

TABLE I
EXPERIMENTAL TIMELINE AND STAGED VALIDATION MAP. THE TABLE REPORTS THE RESEARCH DECISION FOR EACH STAGE RANGE, NOT ONLY WHETHER THE SOFTWARE EXECUTED.

| Stage range | Goal | Backend | Result and gate decision |
|---|---|---|---|
| 01–03 | Minimal architecture feasibility: Transformer reference core, Memory/State/Rule paths, CivilizationBlock prototype, and PyTorch training closure. | NumPy / PyTorch | Passed. Structured paths entered computation and the PyTorch backend produced logits, hidden states, traces, and decreasing training loss. |
| 04–07 | Hidden-state observation, codebook construction, template leakage diagnosis, and cross-template repair. | PyTorch | Leakage and surface-feature risk were found; cross-template alignment repaired the gate and allowed hidden-state injection planning. |
| 08–11 | Hidden-state injection, Memory/State/Rule fusion, CivilizationBlock replacement, and migration-precondition stress testing. | PyTorch | Passed. Injection controls, fusion gates, trace checks, and multi-seed stress profiles remained valid before harder benchmarks were introduced. |
| 12–17 | Hard logic, chain-state repair, path-dependency ablation, and pre-migration pressure testing. | PyTorch | Failed then repaired. Hard multi-hop and priority-control failures blocked migration; chain-state and path-dependent training restored the required gates. |
| 18–26 | Frozen Qwen3 observation, adapter training, multilayer adapters, surface-group repair, and initial real-task migration. | Frozen Qwen3 | Mixed. Frozen adapter and surface-flip repair passed controlled gates, but multilayer and real-task migration exposed failures that blocked direct expansion. |
| 27–38 | Path-specific repair, context/readout alignment, projected delta readout, fixed-centroid alignment, multiclass group curriculum, and fail-fast gate design. | Frozen Qwen3 | Mixed with preserved failures. Path-specific deltas and projected gates improved, but multiclass Memory and full-hidden gates repeatedly failed before final targeted repairs. |
| 39 | Strict Memory five-candidate group gate and fail-fast audit. | Frozen Qwen3 | Failed as intended. Memory projected accuracy and group success stayed near random; runner stopped before Rule, Conflict, Combined, or full-hidden stages. |
| 40 | Memory delta gradient path and five-candidate label mapping repair. | Frozen Qwen3 | Passed. Direct use of differentiable memory-delta tensors repaired Memory projected accuracy, group success, no-memory drop, and wrong-context drop. |
| 41 | Rule delta repair, rule-conditioned conflict, and Combined group recovery. | Frozen Qwen3 | Passed. Rule, Conflict, and Combined projected gates passed while retaining the Stage 40 Memory gate. |
| 42 | Group full-hidden fixed-centroid integration. | Frozen Qwen3 | Passed. Raw full-hidden fixed-centroid readout reached the gate while projected path gates were retained. |
| 42-M | Multi-seed Stage 42 robustness benchmark over seeds 202/303/404. | Frozen Qwen3 | Passed. Fixed-centroid after was $1.0000 \pm 0.0000$, wrong-context drop was $0.8000 \pm 0.0000$, and Qwen3 remained frozen in all seeds. |
| BM-1 | Matched MLP adapter, context-token-only, and task-text-only comparison under the Stage 42 sample/readout contract. | Frozen Qwen3 | Completed over three seeds. MLP matched Civilization at 1.00 accuracy; context-token-only reached 0.9611 and task-text-only remained at 0.20. No predictive advantage was established. |
| BM-2 | Parameter-matched LoRA and prefix-tuning comparison under the same three-seed, 440-step contract. | Frozen Qwen3 | Completed. Both methods matched Civilization at 1.00 accuracy and 0.80 wrong-context drop, confirming a ceiling in the current benchmark. |

### *D. Benchmarks*

The PyTorch backend uses controlled logic tasks, multi-hop reasoning tasks, counterfactual memory swaps, surface-invariant label flips, and stress profiles including obfuscated, noisy-context, and conflicting-context variants. The Qwen3 backend uses binary diagnostics, rule-conditioned conflict diagnostics, and five-candidate groups for Memory necessity, Rule necessity, Memory-Rule conflict, and Combined groups. Each group is evaluated under full context, target-path ablation, irrelevant-path ablation, wrong-context substitution, adapter-disabled, and zero-scale conditions where applicable.

### *E. Baseline Taxonomy and Comparison Protocol*

The comparison protocol separates paired controls, matched competitors, and still-unrun retrieval/agent comparisons. This distinction is necessary because the main experiments were designed for path attribution. Adapter-disabled evaluation preserves the frozen Qwen input and probe protocol but prevents the adapter from writing a residual; it is therefore a paired causal control, not an independently optimized base-Qwen baseline. The full baseline taxonomy is reported in the supplementary material.

The same restriction applies to empty-context and wrong-context conditions: they test whether the latent update uses the supplied structured evidence, but they do not instantiate RAG because no retriever is queried and no evidence is selected from a corpus. The matched rounds hold the Qwen checkpoint, sample contract, candidate options, 440-step training budget, and raw final-hidden fixed-centroid readout constant. The MLP comparator uses layers 16 and 24 with 3,739,650 parameters. LoRA uses rank 13 on all 28 layers' q/k/v/o projections, and prefix tuning learns 65 K/V positions per layer; each has 3,727,360 parameters, within 0.3423% of Civilization's 3,740,164 optimized parameters. Context-token-only serializes the same grounded Memory/Rule/State content with no trainable parameters. Task-text-only omits that content and serves

as the five-candidate no-context lower bound. All methods use seeds 202, 303, and 404; sample-contract hashes match across rounds, the maximum input length is 135 tokens under a 160-token limit, and no truncation occurs.

Artifact paths, runner entry points, per-seed records, and fail-fast gate definitions are provided in the supplementary material.

## VIII. Results

The results are reported in three layers. The controlled PyTorch backend tests whether the architecture is executable, trainable, and path-dependent under fully controlled synthetic settings. The frozen Qwen3 backend tests whether the same structured paths can be attached to a real decoder without modifying base model weights. Negative results are reported separately because several stage failures directly shaped the final protocol.

### A. Controlled PyTorch Results

Table II summarizes the controlled backend. The codebook experiments first showed that hidden states can be clustered, but the initial result was not accepted as semantic evidence because template features were too predictive. After cross-template alignment, canonical, synonym, perturbed, and masked-keyword variants all reached 1.00 nearest-centroid accuracy with zero masked-keyword drop. Hidden-state injection then passed the zero-equivalence and norm gates: alpha-zero injection was exactly equivalent to baseline, target hit rate remained 1.00, and wrong-label drift remained low. Memory/State/Rule fusion preserved the same gates while adding hard-rule block and trace checks.

The hard-logic track then tested whether the controlled model survived more difficult reasoning conditions. The final chain-state benchmark used 10 seeds, 3 sequence lengths, 2 model sizes, and 12 scenarios. Average scenario accuracy was near 1.00; the worst reported scenario was two-hop logic at 0.9991, with no NaN, trace, masked-keyword, or counterfactual-split failures. This result is reported as controlled evidence only; it does not imply public benchmark transfer.

Table III and Fig. 6 report the path-dependency stress test. The full model retained 0.985 average accuracy across 1080 training runs, while structure-only evaluation collapsed to the five-class random baseline of 0.20. Targeted ablations caused selective drops: disabling Memory mainly damaged memory-required and counterfactual-memory scenarios, disabling Rule mainly damaged rule-priority scenarios, and disabling State mainly damaged state-disambiguation scenarios. The stress profiles also show that the architecture is not only fitting one clean template: obfuscated, noisy-context, and conflicting-context profiles retained high full-model accuracy.

### B. Frozen Qwen3 Results

Table IV summarizes the frozen Qwen3 route. The hidden-state baseline exported all 29 hidden-state groups from Qwen3-0.6B and identified layer 16 as a robust adapter target, while keeping the model weights unchanged. The first single-layer adapter was intentionally not treated as a final result: a later medium run reported only 0.20 main-layer accuracy, demonstrating that a trainable adapter hook alone does not solve the problem. The multilayer adapter and surface-flip repair established that adapter-only training can work under frozen-base constraints, but subsequent real-task migration failed its research gates. This forced the project toward path-specific residuals and explicit readout/audit protocols.

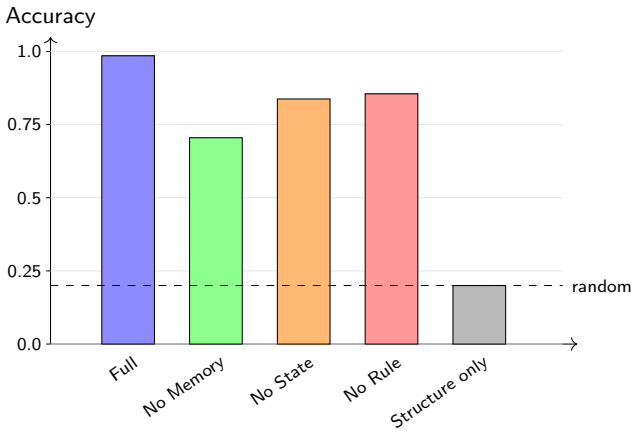


Fig. 6. Controlled path-dependency stress summary. Full-context accuracy remains high, targeted path ablations produce controlled degradation, and the structure-only baseline falls to random chance.

The path-specific adapter made the failure more diagnosable. In the early path-specific binary diagnostic, Memory-only and Rule-only answer readouts were near random (0.4625 and 0.425), even though the conflict diagnostic reached 1.00. Projected delta readout repaired the Memory-only and Rule-only binary cases to 1.00, but the combined binary diagnostic remained at 0.475 and raw delta readout stayed weak. Fixed-centroid alignment later raised the average full-hidden fixed-centroid readout from 0.7375 to 0.9406 without regressing projected gates, showing that projected deltas and raw hidden readout must be audited separately.

Stages 40–42 are the main frozen-model evidence. Stage 39 failed under strict fail-fast gates: Memory delta-to-option accuracy and group success were 0.20 and 0.00. Stage 40 repaired this by reading the differentiable *memory-delta tensor* directly; Memory delta-to-option accuracy, projected accuracy, and group success all reached 1.00, with no-memory and wrong-context drops of 0.80. Stage 41 restored Rule and Conflict groups while retaining the Stage 40 Memory gate. Stage 42 then moved the signal from projected path readout into raw full hidden states. In the original seed-202 run, the fixed-centroid average increased from 0.2074 to 1.00 with projected gates retained and hidden-norm ratio 1.1359. The follow-up Stage 42 multi-seed robustness benchmark repeated the same raw full-hidden integration gate over seeds 202, 303, and 404. All three seeds passed. As shown in Table VI, fixed-centroid-after accuracy was $1.0000 \pm 0.0000$, wrong-context drop was $0.8000 \pm 0.0000$, and hidden-norm ratio was $1.1270 \pm 0.0091$.

TABLE II
CONTROLLED PYTORCH BACKEND SUMMARY. METRICS ARE TAKEN FROM STAGED ARTIFACT SUMMARIES AND REPORT RESEARCH GATES RATHER THAN ONLY SOFTWARE TESTS.

| Experiment | Evidence | Main value | Decision |
|---|---|---|---|
| Codebook generalization | Cross-template alignment over canonical, synonym, perturbed, and masked-keyword variants. | All four trained-mean variants: 1.00; masked-keyword drop: 0.00. | Passed after initial leakage repair. |
| Hidden-state injection | Codebook-centroid intervention with alpha-zero, wrong-label, strong-alpha, and norm gates. | Target hit: 1.00; zero equivalence: 1.00; wrong-label drift: 0.045. | Passed; baseline was saturated, so the gate was non-regression plus audit controls. |
| Memory/State/Rule fusion | Full fusion, memory-guided, state-guided, rule-gated, blocked, and wrong-label modes. | Full fusion target hit: 1.00; hard-rule block: 1.00; wrong-label drift: 0.0075. | Passed with trace and hidden-norm checks. |
| Chain-state alignment | 10 seeds, 3 sequence lengths, 2 model sizes, and 12 hard scenarios. | Worst scenario: two-hop logic 0.9991; chain-step and final target accuracy: 1.00. | Passed controlled hard benchmark. |
| Path dependency | Full, no-memory, no-state, no-rule, and structure-only modes over path-dependent tasks. | Full: 1.00; structure-only: 0.20; target-path drops up to 0.80. | Passed native path-dependency benchmark. |

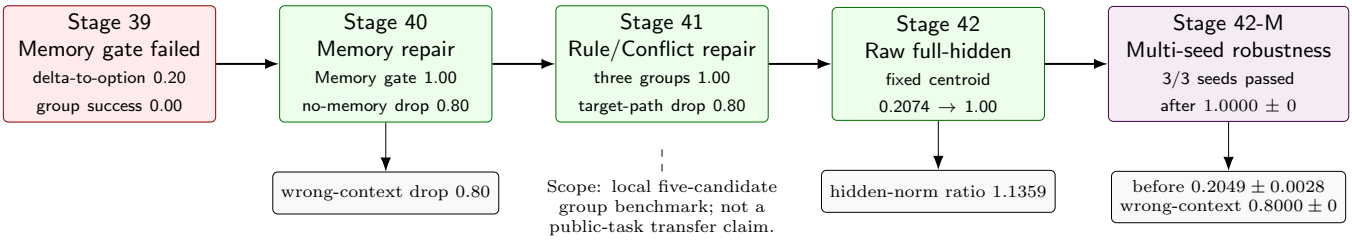


Fig. 7. Stage 39–42-M gate progression. Stage 39 fails the Memory gate; Stage 40 repairs Memory; Stage 41 restores Rule/Conflict; Stage 42 validates raw full-hidden fixed-centroid recovery; Stage 42-M repeats the gate over three seeds under the same local five-candidate benchmark contract.

TABLE III
CONTROLLED PATH-DEPENDENCY STRESS RESULTS ACROSS 1080 TRAINING RUNS.

| Mode | Avg. accuracy | Avg. drop |
|---|---|---|
| Full | 0.9854 | 0.0000 |
| No Memory path | 0.7047 | 0.2807 |
| No State path | 0.8379 | 0.1474 |
| No Rule path | 0.8553 | 0.1301 |
| Structure only | 0.2000 | 0.7854 |

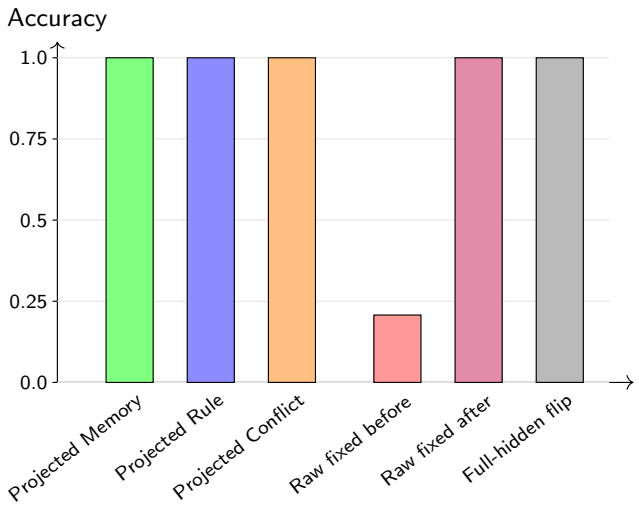


Fig. 8. Projected versus raw full-hidden readout. Projected path readout can pass before raw fixed-centroid readout is aligned; Stage 42 closes this gap for the reported five-candidate group benchmark.

### C. Matched Baseline Results

Table VII shows a clear ceiling effect. The Civilization path, matched MLP adapter, LoRA, and prefix tuning all reach $1.0000 \pm 0.0000$ held-out accuracy and $0.8000 \pm 0.0000$ wrong-context drop. Context-token-only reaches $0.9611 \pm 0.0000$, only 0.0389 below the trained methods, whereas task-text-only remains at the five-candidate chance level of $0.2000 \pm 0.0000$. The current benchmark therefore does not show a predictive accuracy advantage for Civilization over a generic adapter, standard parameter-efficient adaptation, or direct token context.

The ablation columns test different interventions. Civilization disables a native Memory or Rule residual path, producing mean drops of 0.8000 and 0.8019. The MLP, LoRA, prefix, and context-token methods instead remove information from their serialized token context; those values are not mechanism-equivalent effect sizes. They support the narrower conclusion that Civilization exposes separate native intervention points, not that it is more accurate.

Under the recorded CUDA peak-allocated metric, Civilization uses $2.856 \pm 0.509$ GB, compared with $5.277 \pm 0.015$ GB for the matched MLP, $8.118 \pm 0.021$ GB for LoRA, and $9.001 \pm 0.017$ GB for prefix tuning. This is a configuration-specific resource observation: the implementations use differ-

TABLE IV
FROZEN QWEN3 BACKEND SUMMARY. QWEN3-0.6B WEIGHTS REMAIN FROZEN IN ALL LISTED STAGES; ONLY ADAPTER/READOUT COMPONENTS ARE TRAINED WHERE APPLICABLE.

| Stage family | Evidence | Main value | Decision |
|---|---|---|---|
| Hidden-state baseline | Exported Qwen3 hidden states from all 29 hidden-state groups and evaluated layer/pooling candidates. | Best accuracy: 1.00; layer 16 last-pooling robust score: 0.8879. | Passed; allowed frozen adapter training. |
| Single-layer adapter | Frozen layer-16 adapter under engineering gates. | Later medium run main-layer accuracy: 0.20. | Engineering passed, research gate failed. |
| Multilayer adapter | Dual layer 16/24 adapter with frozen Qwen3 and path ablations. | Final accuracy: 1.00; seed std: 0.00. | Passed controlled dependency gate. |
| Path-specific adapter | Independent Base, Memory, Rule, and State residual paths. | Memory-only: 0.4625; Rule-only: 0.425; Conflict: 1.00. | Structure worked, readout gate failed. |
| Projected readout | Projected delta readout over path-specific residuals. | Memory-only: 1.00; Rule-only: 1.00; Combined: 0.475. | Partially passed; led to curriculum repair. |
| Fixed-centroid alignment | Raw full-hidden fixed-centroid alignment after projected gates. | Average before: 0.7375; after: 0.9406. | Passed binary alignment gate. |
| Stages 40–42 | Five-candidate Memory, Rule/Conflict, and full-hidden integration. | Stage 42 multi-seed fixed-centroid after: $1.0000 \pm 0.0000$; hidden-norm ratio: $1.1270 \pm 0.0091$. | Main frozen-model evidence, 3 seeds. |
| Matched baselines | MLP adapter, LoRA, prefix tuning, context-token-only, and task-text-only under a shared sample/readout contract. | Trained methods: 1.00; context tokens: 0.9611; task text: 0.20. | No predictive superiority; native path auditability remains the supported distinction. |

TABLE V
STAGE 40–42 GATE RESULTS ON THE FIVE-CANDIDATE GROUP BENCHMARK.

| Stage | Gate | Value |
|---|---|---|
| 39 | Memory delta-to-option | 0.20 |
| 39 | Memory group success | 0.00 |
| 40 | Memory delta-to-option | 1.00 |
| 40 | Memory group success | 1.00 |
| 40 | No-memory drop | 0.80 |
| 40 | Wrong-context drop | 0.80 |
| 41 | Memory group success | 1.00 |
| 41 | Rule group success | 1.00 |
| 41 | Conflict group success | 1.00 |
| 42 | Fixed-centroid before | 0.2074 |
| 42 | Fixed-centroid after | 1.00 |
| 42 | Hidden-norm ratio | 1.1359 |

TABLE VI
STAGE 42 MULTI-SEED RAW FULL-HIDDEN FIXED-CENTROID ROBUSTNESS.

| Metric | Mean | Std. | $N$ |
|---|---|---|---|
| Fixed-centroid before | 0.2049 | 0.0028 | 3 |
| Fixed-centroid after | 1.0000 | 0.0000 | 3 |
| Memory fixed-centroid | 1.0000 | 0.0000 | 3 |
| Rule fixed-centroid | 1.0000 | 0.0000 | 3 |
| Conflict fixed-centroid | 1.0000 | 0.0000 | 3 |
| Combined fixed-centroid | 1.0000 | 0.0000 | 3 |
| Wrong-context drop | 0.8000 | 0.0000 | 3 |
| Hidden-norm ratio | 1.1270 | 0.0091 | 3 |

ent forward structures and batch construction, and Civilization runtime is assembled from staged historical runs. No general runtime or memory-complexity superiority is inferred.

### *D. Failure-Driven Staged Validation*

The project treats failed gates as design constraints; the full negative-result ledger and public-task control table are reported in the supplementary material. This is important because a late successful gate would be misleading if earlier failures were hidden. Several failures passed engineering checks while failing research checks: Qwen3 remained frozen, artifacts were written, and disabled/zero-scale controls often behaved correctly, yet the required path dependency was absent. Those runs were therefore used to restrict later claims rather than to support the final result.

The earliest Qwen3 failure was a hook sufficiency failure. Single-layer medium adapter variants remained near the five-candidate chance level even though model-integrity and generation-sanity checks passed. This ruled out the weak claim that adding a trainable hidden-state hook is itself evidence of Civilization Architecture. The later architecture therefore required multilayer placement, separate Base/Memory/Rule/State residuals, and trace-exported path deltas.

The next failures separated three different questions that were initially entangled. Real-task migration showed that public and semi-real transfer was not established: the Stage 27B full matrix recorded no meaningful improvement over the adapter-disabled condition on RTE, CB, or BoolQ. Memory/Rule necessity repair then showed that improving a full-hidden fixed-centroid probe does not guarantee answer-option direction or path necessity. Path-specific adapter repair further showed that a nonzero Memory or Rule delta is not enough unless the delta is readable in the intended option space. These failures led to the projected-delta readout and to the rule that projected evidence and raw full-hidden evidence must be audited separately.

Stage 39 is the clearest example of failure-driven validation. Its Memory context was already separable, but the Memory residual path failed to carry the decision: Memory delta-to-option accuracy was 0.20, group success was 0.00, no-memory drop was 0.00, and wrong-context drop was 0.00.

TABLE VII
MATCHED FROZEN-QWEN3 BASELINE RESULTS OVER SEEDS 202, 303, AND 404. ACCURACY AND WRONG-CONTEXT VALUES USE THE SAME HELD-OUT FIVE-CANDIDATE RAW FULL-HIDDEN FIXED-CENTROID PROTOCOL. MEMORY/RULE DROPS ARE NATIVE PATH ABLATIONS ONLY FOR CIVILIZATION; FOR ALL TOKEN-CONTEXT COMPETITORS THEY ARE INFORMATION-CHANNEL REMOVALS AND ARE NOT MECHANISM-EQUIVALENT EFFECT SIZES.

| Method | Accuracy | Wrong-context drop | Memory drop | Rule drop | Trainable params. | Peak CUDA alloc. (GB) |
|---|---|---|---|---|---|---|
| Civilization path | $1.0000 \pm 0$ | $0.8000 \pm 0$ | $0.8000 \pm 0$ | $0.8019 \pm 0.0714$ | 3,740,164 | $2.856 \pm 0.509$ |
| Matched MLP adapter | $1.0000 \pm 0$ | $0.8000 \pm 0$ | $0.2667 \pm 0$ | $0.6000 \pm 0$ | 3,739,650 | $5.277 \pm 0.015$ |
| LoRA | $1.0000 \pm 0$ | $0.8000 \pm 0$ | $0.2667 \pm 0$ | $0.6000 \pm 0$ | 3,727,360 | $8.118 \pm 0.021$ |
| Prefix tuning | $1.0000 \pm 0$ | $0.8000 \pm 0$ | $0.2667 \pm 0$ | $0.6000 \pm 0$ | 3,727,360 | $9.001 \pm 0.017$ |
| Context-token-only | $0.9611 \pm 0$ | $0.7611 \pm 0$ | $0.3611 \pm 0$ | $0.4648 \pm 0$ | 0 | $1.458 \pm 0$ |
| Task-text-only | $0.2000 \pm 0$ | $0.0000 \pm 0$ | N/A | N/A | 0 | $1.278 \pm 0$ |

The runner therefore stopped before Rule, Conflict, Combined, or full-hidden stages. Stage 40 did not merely rerun the same training; it changed the supervision path by reading the differentiable memory-delta tensor directly and freezing non-Memory subpaths. Only after that gate reached 1.00 accuracy and 0.80 no-memory and wrong-context drops was Stage 41 allowed to restore Rule/Conflict groups.

The final failure class concerns representation survival. Before Stage 42 alignment, projected gates passed but raw full-hidden fixed-centroid recovery was near chance: 0.2074 for seed 202 and $0.2049 \pm 0.0028$ over the later three-seed audit. Stage 42 therefore tests a stricter question than projected readout: whether the path signal remains recoverable from the integrated hidden stream using train-only centroids under wrong-context and ablation controls. The successful multi-seed result closes that local gate, but it does not change the external-validity boundary. The paper still does not claim public benchmark transfer or general real-task superiority.

## IX. ABLATION STUDY

The ablation protocol distinguishes path necessity from ordinary accuracy improvement. A run is not accepted only because the full model answers correctly; it must also satisfy target-path, irrelevant-path, wrong-context, disabled-adapter, zero-scale, and fixed-centroid controls where those controls apply. In the controlled PyTorch stress test, disabling Memory, State, or Rule selectively damages the scenario family that requires that path, while the structure-only configuration falls to the five-class chance baseline. In the Qwen3 five-candidate groups, target-path drops reach 0.80 for Memory, Rule, and Conflict groups, whereas irrelevant-path drops remain controlled.

Wrong-context substitution changes structured Memory or Rule content while preserving the surface task format. The final Qwen3 groups retain a wrong-context drop of 0.80, indicating that the readout follows the structured context rather than only the surface template. Disabled-adapter and zero-scale controls verify that the observed signal is not already present in the frozen base model or in the readout alone: the adapter either returns the original hidden state or writes a zero residual, while Qwen3 trainable parameters and gradients remain zero.

Projected delta readout and raw full-hidden fixed-centroid readout serve different audit roles. Projected readout checks whether a path-specific residual delta carries the expected option direction; fixed-centroid readout checks whether that signal survives integration into the raw hidden stream using train-only centroids. Stage 42 improves fixed-centroid recovery from near-chance before alignment to $1.0000 \pm 0.0000$ after alignment over three seeds while retaining projected gates and wrong-context control.

## X. DISCUSSION

The central result of the staged evaluation is controlled path necessity rather than unconstrained task accuracy. The PyTorch backend shows that Memory, State, and Rule can be implemented as native computational paths whose removal causes selective degradation under synthetic stress profiles. The frozen Qwen3 backend shows a narrower but important real-model instantiation: structured Memory and Rule signals can be injected through path-specific latent adapters while the base decoder remains frozen, and the resulting signals can be audited through projected deltas, fixed train-only centroids, wrong-context substitution, and target-path ablation.

This distinction matters because many external augmentation systems can improve answers without making the source of the decision auditable. A retrieval system may place useful facts in the prompt, an agent controller may choose tools, and a prompt template may express behavioral constraints, but these mechanisms do not by themselves establish that memory, state, or rule signals are represented as separable latent paths. The reported experiments instead ask whether the decision changes when the intended path is disabled and remains stable when irrelevant paths are disabled. In that sense, the main contribution is an architectural and evaluation protocol for latent path attribution.

The matched comparison sharpens this interpretation. A generic MLP adapter, LoRA, and prefix tuning reach the same accuracy and wrong-context drop as Civilization, while direct token context is already near ceiling. The present data therefore reject an accuracy-superiority interpretation. The supported difference is architectural: Civilization provides separately addressable Memory and Rule residual paths with native ablation points. This is a controllability and audit-interface contribution, not a predictive-performance gain. RAG remains outside the comparison because context-token-only uses fixed provided evidence rather than retrieval from a corpus.

The negative stages are important to this interpretation. Early hidden-state codebooks exposed template leakage and therefore could not support semantic claims until cross-template controls were added. Early Qwen3 adapters could

satisfy engineering gates while failing research gates, showing that a hook or trainable adapter is not sufficient evidence of path necessity. Public and semi-real task migration also failed under the current protocol. These failures prevent the paper from claiming that Civilization Architecture already solves general real-world reasoning, but they strengthen the methodological claim that the protocol can reject insufficient explanations.

The frozen Qwen3 results should also be interpreted as an instantiation, not as a universal property of all language models. Qwen3-0.6B provides a concrete decoder hidden stream in which path-specific residuals can be trained and audited. The same conclusion may not hold without modification for other model families, larger context lengths, different tokenizer distributions, mixture architectures, multimodal decoders, or unfrozen training regimes. The current paper therefore treats Qwen3 as evidence that the interface is feasible in one real LLM, not as evidence that every LLM admits the same adapter geometry.

Stage 42 closes an important gap between projected path readout and raw full-hidden readout, but it also exposes a remaining integration risk. The full-hidden fixed-centroid recovery depends on a residual write configuration that has now passed the reported three-seed controlled robustness benchmark. The observed hidden-norm ratio remained within the stage gate across those seeds, but residual scale is still a tuning-sensitive mechanism. It should be treated as an empirical integration parameter that requires longer-context, larger-seed, cross-task, and cross-model stress testing before it can be recommended as a stable default.

Finally, Civilization Architecture is broader than the present empirical slice. The architecture includes long-term memory maintenance, state evolution, rule revision, audit isolation, and human-governed update loops as design components. The experiments in this paper validate the more limited substrate needed for those components: structured pathways can enter hidden-state computation and can be ablated and audited. They do not demonstrate completed lifelong learning, autonomous memory consolidation, or production governance.

## XI. Limitations

The current evidence has several limitations.

### A. Threats to Validity

Internal validity is addressed through fail-fast gates, preserved failure artifacts, adapter-disabled and zero-scale controls, train-only centroids, wrong-context substitutions, and frozen-weight integrity checks. The remaining internal risk is that the final controlled benchmark may still contain latent regularities that make all trained methods reach a ceiling. Construct validity is limited by the use of fixed-centroid and option-readout metrics: these metrics are appropriate for path attribution, but they are not substitutes for open-ended generation quality, retrieval quality, or real user-task success. External validity is limited to the reported controlled PyTorch tasks and the frozen Qwen3-0.6B instantiation. Implementation validity is strengthened by runner-level artifacts, seed manifests, sample-contract checks, and checkpoint audits, but independent reimplementation on another model family remains future work.

### B. Controlled Benchmark Scope

The strongest claims concern controlled path necessity. The PyTorch backend and the Qwen3 group benchmarks were designed to isolate Memory, State, and Rule paths under synthetic or local tasks. This design is appropriate for causal diagnosis, but it does not establish public benchmark transfer, broad language understanding gains, or superiority over retrieval-augmented and agentic systems on open-ended workloads. Public real-task migration remains an unresolved result in the current project.

### C. Stage 42 Robustness Scope

The Stage 42 full-hidden fixed-centroid result has been replicated over three seeds (202, 303, and 404), and all three seeds pass the reported controlled gate. This removes the earlier single-seed limitation for the local five-candidate robustness benchmark. It does not, however, establish distributional robustness over arbitrary seeds, larger task families, longer contexts, public benchmarks, or production workloads. The current multi-seed claim is therefore restricted to the reported controlled benchmark.

### D. Competitive Baseline Coverage

The baseline suite now includes paired causal controls and matched comparisons with task-text-only input, context-token-only input, a generic bottleneck adapter, LoRA, and prefix tuning. All trained matched methods saturate the current benchmark, so it cannot resolve predictive differences among them. The three seeds share the same data and split and vary initialization and training order; zero variance therefore does not establish distributional robustness. The benchmark also does not include a retriever-backed RAG system, standard generative task metrics, or a harder held-out distribution with paraphrased context, answer-phrase masking, long noise, or shifted conflict rules. The recorded RTE, CB, and BoolQ results compare the full adapter with an adapter-disabled hidden-state probe and fail the public transfer gate; they are not generative prompt baselines. Consequently, this paper does not claim predictive superiority over prompting, retrieval, or parameter-efficient fine-tuning methods.

### E. Model-Family Generality

The frozen real-model instantiation uses Qwen3-0.6B with frozen base weights. The experiments do not show that the same path-specific adapter design, layer choice, centroid geometry, residual scale, or readout protocol transfers to all Transformer language models. The matched LoRA and prefix results apply only to the local implementations and exact configurations reported here; they are not claims about all PEFT implementations. The experiments do not compare against full-parameter fine-tuning, larger Qwen variants, non-Qwen decoders, or multimodal architectures. The paper therefore claims feasibility for one frozen-LLM instantiation and leaves cross-family validation as future work.

### F. Current Training Boundary

The current paper does not pretrain a full native Civilization foundation model and does not update the Qwen3 base model in the frozen instantiation. Trainable parameters are restricted to adapter and readout modules in the Qwen3 experiments, while the controlled PyTorch backend trains a native CivilizationTransformer. This is an experimental boundary, not the intended long-term training definition. The intended training object for later work is the complete Civilization model-system architecture, including the neural core, Memory encoders, State gates, Rule pathways, readout heads, and memory-evolution interfaces. The present results should therefore be interpreted as staged evidence for architecture-level path training, not as evidence that a general-purpose native Civilization foundation model has already been trained.

### G. Residual-Scale and Readout Sensitivity

The transition from projected delta readout to raw full-hidden fixed-centroid readout required explicit alignment and a stronger residual write than earlier projected stages. Although the Stage 42 hidden-norm gate passed across the reported three seeds, the residual-scale setting may interact with context length, option count, task distribution, layer selection, and model scale. Future evaluations should sweep residual scales, report norm and accuracy tradeoffs, and test whether the same configuration remains stable under longer and more heterogeneous inputs.

### H. Lifelong Learning and Governance

Civilization Architecture includes memory maintenance, state evolution, rule revision, audit isolation, and human approval loops as long-term design goals. These components are not completed empirical claims in this paper. The present work validates structured hidden-state pathways and path-specific audit gates; it does not demonstrate autonomous lifelong learning, persistent graph-memory governance, safety approval workflows, or deployment-ready operation.

## XII. Conclusion

This paper presented Civilization Architecture, a staged model-system architecture for moving structured Memory, State, and Rule signals from prompt-level context into hidden-state computation. The model is defined as the integrated computation graph, not as a standalone LLM plus external tools. The controlled PyTorch backend validates native Memory/State/Rule fusion, tracing, hidden-state intervention, and selective path degradation under stress tests. The frozen Qwen3 instantiation provides a real-model test of the same boundary: path-specific adapters inject structured latent residuals into the neural core's hidden stream while the base decoder remains unchanged for auditability.

The empirical claim is deliberately bounded. The results support controlled path necessity, fail-fast ablation auditing, projected path readout, and three-seed raw full-hidden fixed-centroid integration. They do not establish predictive superiority: matched MLP, LoRA, and prefix-tuning baselines reach the same accuracy ceiling, and context-token-only is near ceiling. The demonstrated increment is a native path-level intervention and audit surface. The results do not establish public benchmark transfer, lifelong learning, or a newly pretrained native Civilization foundation model. Future work should train larger native Civilization architectures, evaluate a preregistered harder held-out protocol, test retriever-backed RAG and generative metrics, broaden public and semi-real tasks, and implement graph-backed memory update and governance protocols.

## Appendix A
## Reproducibility Checklist

This appendix records the reproducibility information accompanying this public preprint. The companion supplementary material expands this checklist with the full stage runner ledger, artifact manifest, Stage 42 per-seed metrics, fail-fast gates, frozen-Qwen3 integrity checks, and baseline definitions.

### A. Primary Artifact Manifest

The paper tables are derived from the following artifact summaries:

- Controlled codebook generalization: `experiments/civilization_transformer_torch/artifacts/codebook_generalization/alignment_until_pass/iteration_01/summary.json`.
- Hidden-state injection: `experiments/civilization_transformer_torch/artifacts/hidden_injection/summary.json`.
- Memory/State/Rule fusion injection: `experiments/civilization_transformer_torch/artifacts/fusion_injection/summary.json`.
- Chain-state full benchmark: `experiments/civilization_transformer_torch/artifacts/chain_state_full_benchmark/summary.json`.
- Path-dependency stress test: `experiments/civilization_transformer_torch/artifacts/path_dependency_stress_test/summary.json`.
- Frozen Qwen3 hidden-state baseline: `experiments/civilization_transformer_qwen3/artifacts/hidden_state_baseline_medium/summary.json`.
- Public-task adapter-disabled controls and failed transfer gate: `experiments/civilization_transformer_qwen3/artifacts/real_task_migration/summary.json`.
- Round-1 matched MLP, context-token, and task-text baselines: `experiments/civilization_transformer_qwen3/artifacts/matched_baselines_round1/summary.json`.
- Round-2 matched LoRA and prefix-tuning baselines: `experiments/civilization_transformer_`

qwen3/artifacts/matched_baselines_round2_peft/summary.json.

- Stage 40 Memory repair: experiments/civilization_transformer_qwen3/artifacts/memory_delta_gradient_diagnostic/summary.json.
- Stage 41 Rule/Conflict recovery: experiments/civilization_transformer_qwen3/artifacts/rule_conflict_group_recovery/summary.json.
- Stage 42 full-hidden integration: experiments/civilization_transformer_qwen3/artifacts/group_full_hidden_centroid_integration/summary.json.
- Stage 42 multi-seed robustness: experiments/civilization_transformer_qwen3/artifacts/stage42_multiseed_robustness/summary.json.
- Supplementary Stage 43 residual-scale stress artifact: experiments/civilization_transformer_qwen3/artifacts/stage43_residual_scale_stress/summary.json.

### B. Runner Entry Points

The main controlled-backend runners are run_chain_state_full_benchmark.py and run_path_dependency_stress_test.py. The main frozen-Qwen3 runners are run_qwen3_memory_delta_gradient_diagnostic.py, run_qwen3_rule_conflict_group_recovery.py, and run_qwen3_group_full_hidden_centroid_integration.py. The Stage 42 multi-seed robustness runner is run_qwen3_stage42_multiseed_robustness.py. These runners write JSON summaries, CSV audits, checkpoints, and failure reports into their corresponding artifact directories.

### C. Frozen-Model Integrity

The Qwen3 backend validates the local model configuration, checks the expected weight SHA256, sets all base-model parameters to non-trainable, disables cache during hidden-state export, and verifies 29 hidden-state groups. The Stage 40–42 summaries report zero Qwen trainable parameters, zero Qwen gradients, and unchanged weights. Checkpoints are rejected if they contain forbidden Qwen state dictionaries.

### D. Known Reproducibility Boundary

The Stage 42 full-hidden fixed-centroid result is replicated over seeds 202, 303, and 404 under CUDA bfloat16 in the local five-candidate group benchmark. The standalone supplement also records a Stage 43 residual-scale stress artifact over five seeds and two sequence lengths. These artifacts support the reported controlled robustness claim, but they do not prove public benchmark transfer, arbitrary-seed stability, long-context behavior, or cross-model generality. Real-task migration attempts are preserved as negative results.

## References


[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W.-t. Yih, T. Rocktaschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, 2020.

[3] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark *et al.*, "Improving language models by retrieving from trillions of tokens," in *International Conference on Machine Learning*, 2022.

[4] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, 2023.

[5] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations*, 2023.

[6] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[7] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[8] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in Neural Information Processing Systems*, 2015.

[9] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International Conference on Machine Learning*, 2019.

[10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[11] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Annual Meeting of the Association for Computational Linguistics*, 2021.

[12] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Conference on Empirical Methods in Natural Language Processing*, 2021.

[13] X. Liu, K. Ji, Y. Fu, W. L. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *arXiv preprint arXiv:2110.07602*, 2021.

[14] R. K. Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," in *Advances in Neural Information Processing Systems*, 2021.

[15] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," in *International Conference on Learning Representations*, 2019.

[16] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon *et al.*, "Constitutional ai: Harmlessness from ai feedback," *arXiv preprint arXiv:2212.08073*, 2022.

[17] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt, "Interpretability in the wild: A circuit for indirect object identification in gpt-2 small," in *International Conference on Learning Representations*, 2023.

[18] C. Burns, H. Ye, D. Klein, and J. Steinhardt, "Discovering latent knowledge in language models without supervision," *arXiv preprint arXiv:2212.03827*, 2022.

[19] A. Zou, L. Phan, S. Wang, D. Duenas, M. Lin, M. Andriushchenko, J. Z. Kolter, M. Fredrikson, and D. Hendrycks, "Representation engineering: A top-down approach to ai transparency," *arXiv preprint arXiv:2310.01405*, 2023.

[20] Qwen Team, "Qwen3: Think deeper, act faster," https://qwenlm.github.io/blog/qwen3/, 2025, accessed: 2026-06-30.