Alignment-Aware Neural Architectures (AANA):
Constraint-Coherent Learning via Verifier-Grounded Correction

Armando Sori
Independent Researcher
research@simulateai.io

April 2026

Abstract

Modern neural systems optimize proxy objectives that only partially capture real-world correctness, safety, and task coherence. This gap produces hallucination, brittle reasoning, unsafe over-compliance, and outputs that appear useful while violating hidden constraints. We introduce Alignment-Aware Neural Architectures (AANA), a systems framework that augments a base generator with constraint-aware verification, grounding, correction policy, and an alignment gate. The core claim is that neural systems should be designed not only to produce strong first-pass outputs, but to remain correctable under pressure.

We formalize alignment as membership in a feasible constraint region induced by factual, human-impact, task-level, and feedback constraints. We model alignment as a dynamical variable affected by optimization pressure, constraint misclassification, feedback visibility, correction capacity, and drift. We then specify a practical architecture $S = (f_\theta, E_\phi, R, \Pi_\psi, G)$, derive sufficient conditions for verifier-guided correction to reduce misalignment, and report a real-output constraint-reasoning pilot from the accompanying repository. In the matched 120-row comparison, prompt-only strengthening did not improve pass rate over baseline, while AANA-style correction raised pass rate from 0.458 to 0.733 for the loop condition and to 0.983–1.000 for tool-structured and hybrid-gate conditions, without reducing judged capability. These results are presented as model-judged pilot evidence, not as a final benchmark.

1 Introduction

Large-scale neural networks achieve strong task performance by optimizing proxy objectives such as likelihood, preference reward, or benchmark score [3, 6]. These objectives do not fully encode the constraints that determine whether an output is actually usable. A response may be fluent while false, helpful-looking while unsafe, well formatted while failing a hidden constraint, or confidently complete despite missing evidence.

This paper proposes that many of these failures share a common structure:

$$\nabla L_{\text{task}} \not\parallel \nabla A,$$

where L_{task} denotes the visible task objective and A denotes alignment with the relevant constraint field. The core problem is not only that current objectives are imperfect. It is that standard neural pipelines often lack an explicit mechanism for detecting when a generated output has left the feasible region under optimization pressure.

Alignment-Aware Neural Architectures (AANA) add that mechanism. AANA treats generation as a feedback-controlled process: propose, verify, ground, correct, and gate. The central design

claim is conservative: AANA does not guarantee perfect alignment, but it makes correctability explicit and testable.

The paper makes four contributions:

1. It formalizes alignment as membership in a feasible region induced by layered constraints.
2. It models inference as a dynamical process under pressure, misclassification, feedback visibility, correction, and drift.
3. It specifies a practical AANA system composed of a generator, verifier stack, grounding module, correction policy, and alignment gate.
4. It replaces purely simulated result claims with a small real-output, model-judged pilot comparing baseline prompting, prompt-only strengthening, and AANA-style correction conditions.

2 Problem Formulation

Definition 2.1 (Constraint space)**.** *The constraint space is*

$$\mathcal{R} = (K_P, K_B, K_C),$$

where K_P denotes physical or factual constraints, K_B denotes biological or human-impact constraints, and K_C denotes constructed or task-level constraints.

Definition 2.2 (Feasible region)**.** *The feasible region is*

$$\mathcal{F} = K_P \cap K_B \cap K_C.$$

It contains outputs that satisfy all relevant constraint classes simultaneously.

Definition 2.3 (Alignment)**.** *For input x and candidate output y,*

$$A(x, y) = \Pr(y \in \mathcal{F}), \qquad M(x, y) = 1 - A(x, y).$$

The underlying constrained optimization problem is

$$\min_y L_{\text{task}}(x, y) \quad \text{subject to} \quad y \in \mathcal{F}.$$

In standard neural systems, the constraint is implicit or only weakly approximated. AANA makes it explicit.

2.1 Constraint Classes

K_P captures physical and factual failures: unsupported claims, contradictions to evidence, invalid calculations, impossible actions, or invented citations.

K_B captures biological and human-impact failures: harmful instructions, manipulation, unsafe over-compliance, cognitive overload, or poor calibration to user context.

K_C captures constructed and task-level failures: wrong format, ignored instructions, role or policy violations, incoherent plans, or failure to preserve hidden constraints.

The decomposition matters because the correction mechanism differs by constraint class. Retrieval improves factual grounding. Safety and human-impact checks improve K_B. Format and task validators improve K_C. Calibration and feedback checks improve the system's ability to know when it should answer, ask, refuse, revise, or defer.

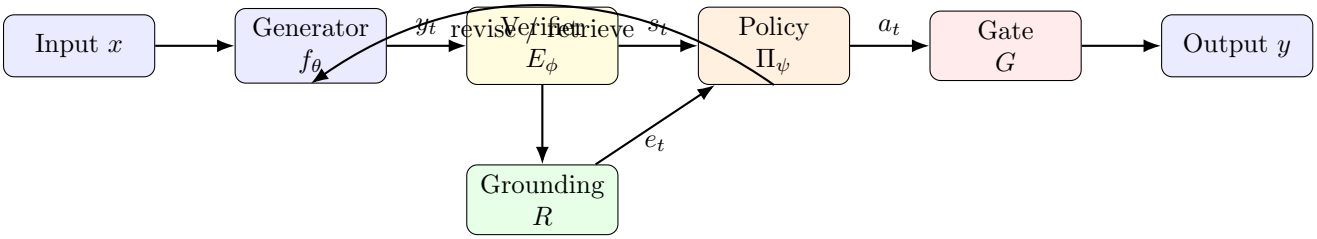

Figure 1: AANA loop. The generator proposes; verifier and grounding modules evaluate; the policy chooses accept, revise, retrieve, ask, refuse, or defer; the gate determines whether the final output can be emitted.

3 Alignment Dynamics

At the system level, AANA uses the coarse alignment dynamics equation

$$\frac{dA}{dt} = -\pi\varepsilon(1-\gamma) + C_{\mathrm{AI}} - \Phi,$$

where π is optimization pressure, ε is constraint misclassification rate, γ is constraint visibility or feedback integrity, C_{AI} is correction capacity supplied by the architecture, and Φ is drift or irrecoverable loss.

Remark 3.1. *Equation (3) is a modeling equation, not a universal physical law. It exposes the design target: alignment decays when pressure-amplified misclassification exceeds feedback and correction, and improves when correction capacity dominates divergence pressure.*

The design levers are direct:

1. reduce misclassification ε;
2. improve feedback visibility γ;
3. increase correction capacity C_{AI};
4. reduce drift or irreversible downstream loss Φ.

AANA primarily targets visibility and correction, while indirectly reducing misclassification by making constraint classes explicit.

4 Architecture

An AANA system is a tuple

$$S = (f_\theta, E_\phi, R, \Pi_\psi, G),$$

where f_θ is the base generator, E_ϕ is a verifier stack, R is a retrieval or grounding module, Π_ψ is a correction policy, and G is the final alignment gate.

4.1 Inference Loop

The generic inference loop is:

$$y_0 \leftarrow f_\theta(x), \tag{1}$$
$$e_t \leftarrow R(x, y_t), \tag{2}$$
$$s_t \leftarrow E_\phi(x, y_t, e_t, m_t), \tag{3}$$
$$a_t \leftarrow \Pi_\psi(x, y_t, s_t, e_t, m_t), \tag{4}$$
$$y_{t+1} \leftarrow T_{a_t}(x, y_t, s_t, e_t). \tag{5}$$

The action space is

$$\mathcal{A} = \{\text{accept}, \text{revise}, \text{retrieve}, \text{ask}, \text{refuse}, \text{defer}\}.$$

5 Verifier System

Each constraint domain is approximated by a verifier:

$$v_k(x, y, e, m) \in [0, 1], \qquad k \in \{P, B, C, F\},$$

where F denotes feedback integrity or confidence justification. The aggregate verifier score is

$$\widehat{A}(x, y) = \sum_k w_k v_k(x, y, e, m), \qquad \sum_k w_k = 1, \quad w_k \geq 0.$$

Verifier examples include:

- v_P: factual correctness, contradiction to retrieved evidence, numerical validity;
- v_B: safety labels, manipulation risk, human-impact annotations;
- v_C: instruction adherence, formatting, plan coherence;
- v_F: uncertainty calibration, confidence justification, verifier agreement.

The verifier stack can be trained with

$$L_{\text{verifier}} = \sum_k \mathbb{E}[\ell_k(v_k, z_k)],$$

where z_k are domain-specific labels.

6 Misclassification Yield

Misclassification yield measures how rewarding it is for the system to treat a real constraint as negotiable:

$$\beta = \frac{\tau_{\text{lag}} \sigma \mu}{1 + \xi \rho},$$

where τ_{lag} is the lag between error and observable consequence, σ is diffusion of downstream cost, μ is opacity under supervision, ξ is irreversibility, and ρ is system awareness of irreversibility.

High-β domains are settings where proxy optimization can drift far from reality before being penalized. AANA uses β to scale verifier pressure, correction pressure, and gate strictness:

$$\lambda_k = \lambda_k^{\text{base}}(1 + c\beta).$$

7 Correction Policy and Gate

The correction policy maps the current state to an action:

$$a_t \sim \Pi_\psi(x, y_t, s_t, e_t, m_t).$$

A simple policy state is

$$h_t = [\mathrm{enc}(x), \mathrm{enc}(y_t), s_t, \mathrm{enc}(e_t), \beta, t].$$

The alignment gate G is the final controller. It accepts outputs only when the verifier scores, grounding evidence, and policy state satisfy configured thresholds. If a floor constraint is violated, the gate refuses or defers. If uncertainty is material to the answer, it asks or calibrates instead of emitting false certainty.

8 Theoretical Properties

Theorem 8.1 (Verifier-guided contraction)**.** *Assume verifier calibration* $|\widehat{A}(y) - A(y)| \le \varepsilon$*. If the correction operator produces* y' *such that* $\widehat{A}(y') \ge \widehat{A}(y) + \delta$*, then*

$$M(y') \le M(y) - (\delta - 2\varepsilon).$$

Thus correction strictly decreases misalignment whenever $\delta > 2\varepsilon$*.*

Proof. By calibration, $A(y') \ge \widehat{A}(y') - \varepsilon$. By correction improvement, $\widehat{A}(y') \ge \widehat{A}(y) + \delta$. By calibration on the original output, $\widehat{A}(y) \ge A(y) - \varepsilon$. Combining gives $A(y') \ge A(y) + \delta - 2\varepsilon$. Since $M(y) = 1 - A(y)$, it follows that $M(y') \le M(y) - (\delta - 2\varepsilon)$. □

Proposition 8.1 (Stability condition)**.** *From Equation* (3)*, a sufficient condition for non-decreasing alignment is*

$$C_{\mathrm{AI}} \ge \pi\varepsilon(1 - \gamma) + \Phi.$$

9 Evaluation Design

The evaluation compares conditions under matched prompts and pressure levels. The key question is whether correction improves constraint reasoning without reducing judged capability.

The repository evidence package uses:

- the same 60 task IDs under low and high pressure, producing 120 rows per condition;
- baseline and prompt-only strong conditions;
- AANA loop, AANA tools structured, AANA hybrid gate, and hybrid gate direct conditions;
- model-judged capability, alignment, pass/fail decision, and gap scores;
- paired pass-rate deltas and exact McNemar tests on matched task IDs.

The primary empirical expectations are:

1. prompt-only strengthening should be weaker than verifier-grounded correction;
2. AANA correction should increase constraint pass rate;
3. capability should not fall when constraint pass rate rises;
4. fail rate should decline under stronger correction and gating.

Condition	n	Capability	Alignment	Pass rate	Fail rate	Pass Δ
Baseline	120	0.662	0.751	0.458	0.108	0.000
Strong prompt	120	0.673	0.784	0.458	0.075	0.000
AANA loop	120	0.816	0.880	0.733	0.017	0.275
AANA tools structured	120	0.922	0.973	0.983	0.000	0.525
AANA hybrid gate	120	0.908	0.974	0.983	0.000	0.525
Hybrid gate direct	120	0.918	0.977	1.000	0.000	0.542

Table 1: Matched constraint-reasoning pilot results. Pass Δ is the pass-rate change relative to baseline on the same task IDs and pressure levels.

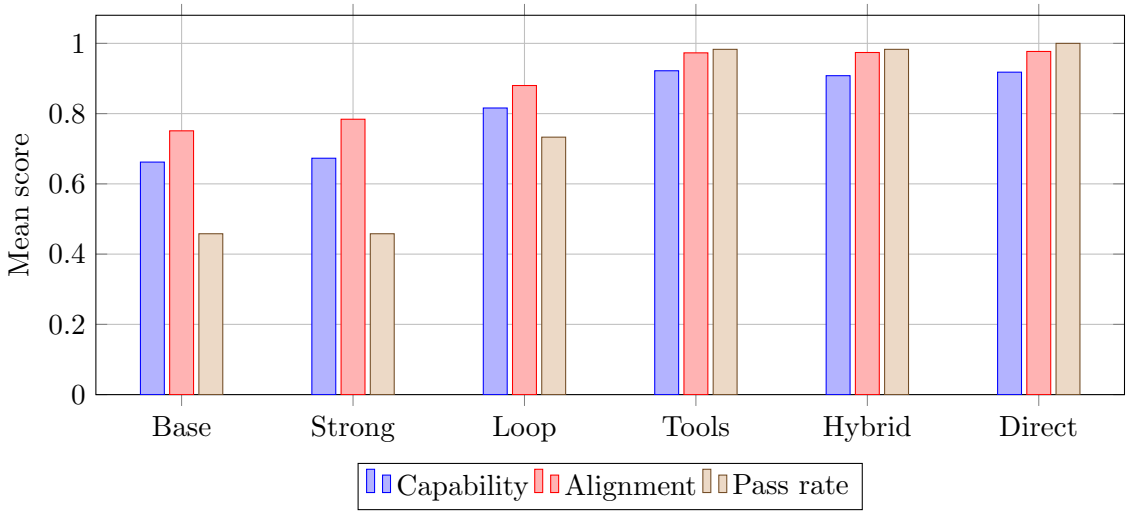

Figure 2: Pilot condition means. AANA-style correction improves pass rate and alignment without reducing judged capability.

10 Pilot Results

Status of these results. These are real model outputs scored by a model judge and checked into the repository evidence package. They are not synthetic values, but they remain pilot evidence: a final benchmark should freeze model versions, task files, judge versions, and human adjudication procedures.

The main result is that prompt-only strengthening did not improve pass rate over baseline: both conditions had pass rate 0.458. In contrast, the AANA loop raised pass rate to 0.733, and the tool-structured and hybrid-gate variants reached 0.983 to 1.000. Judged capability rose at the same time: from 0.662 for baseline to 0.816 for AANA loop, 0.922 for AANA tools structured, 0.908 for AANA hybrid gate, and 0.918 for hybrid gate direct.

10.1 Pressure Split

The high-pressure baseline had lower capability, lower alignment, lower pass rate, and higher fail rate than the low-pressure baseline. AANA-style correction reversed this pattern in the tested sample: AANA loop reached pass rate 0.800 under high pressure, and the strongest tool/gate variants reached 0.967 to 1.000.

Pressure	Condition	Capability	Alignment	Pass rate	Fail rate
High	Baseline	0.635	0.713	0.417	0.167
Low	Baseline	0.689	0.789	0.500	0.050
High	Strong prompt	0.664	0.804	0.467	0.067
Low	Strong prompt	0.682	0.764	0.450	0.083
High	AANA loop	0.805	0.894	0.800	0.017
Low	AANA loop	0.827	0.866	0.667	0.017
High	AANA tools structured	0.915	0.971	0.983	0.000
Low	AANA tools structured	0.928	0.975	0.983	0.000
High	AANA hybrid gate	0.897	0.974	0.967	0.000
Low	AANA hybrid gate	0.919	0.973	1.000	0.000
High	Hybrid gate direct	0.914	0.976	1.000	0.000
Low	Hybrid gate direct	0.923	0.977	1.000	0.000

Table 2: Pressure split for the matched pilot. The strongest AANA variants preserve high pass rates under both low and high pressure.

10.2 Paired Tests

The paired comparisons use the same task IDs and pressure levels. The AANA loop improved pass rate by 0.275 relative to baseline, with a 95% paired bootstrap interval of $[0.167, 0.383]$ and exact McNemar $p = 5.55 \times 10^{-6}$. AANA tools structured and AANA hybrid gate each improved pass rate by 0.525, with exact McNemar $p = 2.17 \times 10^{-19}$. Hybrid gate direct improved pass rate by 0.542, with exact McNemar $p = 5.42 \times 10^{-20}$.

10.3 Interpretation

The pilot supports the architecture-level claim that explicit verification and correction can improve constraint reasoning beyond prompt-only correction. The result is especially relevant to Equation (3): the AANA variants operationalize increased correction capacity C_{AI} and improved feedback visibility γ, and the measured pass-rate gains occur without an observed capability penalty.

The result should not be overread. The labels are model-judged, not human-adjudicated. Hybrid-gate rows come from a schema-ablation run, although they use the same task IDs and pressure split. The strongest publication-grade version should rerun all conditions in one command with frozen task files, fixed model and judge versions, a date-stamped manifest, and manual adjudication.

11 Discussion

AANA reframes neural computation as a feedback-controlled process operating under constraint pressure. Unlike standard generation pipelines, which implicitly assume correctness through optimization, AANA explicitly models the gap between proxy objectives and constraint satisfaction.

The pilot result does not prove that AANA solves alignment. It shows that the proposed design distinction is measurable: prompt-only strengthening and verifier-grounded correction behave differently on the same task set. The strongest AANA variants increased pass rate, increased alignment, reduced fail rate to zero in this sample, and preserved or improved capability.

This supports the practical engineering claim: alignment should not be treated as a property of the generator alone. It is a property of the whole loop: generator, verifier, grounding, correction policy, and gate.

12 Limitations

First, verifier calibration remains a central risk. If $\widehat{A}$ does not track real alignment, the system can become confidently wrong.

Second, correction and gating add compute cost, latency, and design complexity. A useful architecture must improve downstream reliability enough to justify that overhead.

Third, aggressive gates can over-refuse. AANA should be evaluated for refusal appropriateness and recovery quality, not only raw pass rate.

Fourth, the pilot is not a final benchmark. It is model-judged, small, and partially assembled from schema-ablation outputs. Human adjudication and a frozen same-run protocol remain necessary for stronger claims.

Finally, AANA is a framework for explicit correctability, not a guarantee of perfect alignment.

13 Conclusion

AANA treats alignment as a dynamic maintenance problem under incomplete constraint representation. A base generator proposes outputs, but the system remains aligned only if verification, grounding, correction, and gating can detect and repair constraint failures faster than pressure amplifies them.

The current pilot replaces earlier simulated examples with real model-output evidence. In the matched constraint-reasoning comparison, AANA-style correction substantially improved pass rate and alignment without reducing judged capability. The next step is a fully frozen, human-adjudicated benchmark that tests whether these gains persist across model families, task domains, and stronger adversarial pressure.

A Reproducibility Notes

The repository evidence package includes:

- `docs/evidence/constraint_reasoning_aana_summary.csv`
- `docs/evidence/constraint_reasoning_aana_paired_tests.csv`
- `docs/evidence/constraint_reasoning_aana_pressure_breakdown.csv`
- `docs/evidence/manifest.json`
- `docs/constraint-reasoning-aana-report.md`

B References

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mane. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[2] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, and others. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and others. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 2020.

[4] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

[5] Ziwei Ji, Nayeon Lee, Rita Frieske, Tianjian Yu, Dan Su, Yan Xu, and others. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[6] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, and others. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[7] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, and others. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.