Small Models, Strong Contracts

EL-JEFFE 4B as a local-first executive intelligence with deterministic authority and receipts

Hayden Lindley · Glyphd Labs

2026-07-27

Contents

Small Models, Strong Contracts

Abstract

Compact language models are increasingly capable, but a small local model is often judged against a cloud model on the cloud model's preferred task: absorb a large context, reason freely, write a complete answer, manage tools, remember the user, and police its own authority. That comparison confuses model capability with system architecture.

EL-JEFFE 4B is a local-first, mobile-native executive intelligence program built around a strict division of labor. The checkpoint learns semantic compilation, authority, modality, scope, substitution, evidence discipline, routing, abstention, repair, and structured action generation. The deterministic runtime retains personal memory, accepted project truth, permissions, secrets, policy enforcement, tool execution, rollback, receipts, adapter verification, and hardware scheduling.

The model proposes. **Jefe Core** validates. The device executes. Receipts prove what happened.

The existing package is a specification and build-control system, not a trained model release. It contains 241 files spanning contracts, schemas, agent guidance, verification scripts, evaluation suites, design briefs, phase gates, and examples. Phase 0 structural verification passed, including linting, type checking, schema validation, agent guards, and twelve Python tests; all model-quality gates remain deliberately incomplete.

1. The wrong small-model question

The usual question is:

> Can a 4B model act like the best cloud assistant?

The better question is:

> Which responsibilities genuinely require learned judgment, and which should never have been delegated to the model?

A model should not need to own:

- the user's durable memory;
- current project truth;
- OS permissions;
- secrets;
- approval validity;
- tool rollback;
- exact citation verification;
- audit logs;
- cache invalidation;
- hardware scheduling;
- spend limits.

These are better expressed as deterministic state and policy.

A compact model can focus on:

- interpreting messy intent;
- classifying modality and authority;
- extracting constraints;
- building typed graphs;
- selecting the next bounded operation;

- drafting structured proposals;
- recognizing when to stop;
- routing to a specialist;
- repairing malformed output;
- explaining decisions.

2. The checkpoint/runtime boundary

Train into the checkpoint

- Glyph fluency and semantic compression;
- authority, modality, scope, and substitution judgment;
- evidence discipline;
- bounded operational decisions;
- structured action generation;
- routing;
- abstention;
- repair;
- escalation.

Keep outside the checkpoint

- personal and project memory;
- live facts and indexes;
- OS permissions and secrets;
- capability leases;
- deterministic validation;
- policy enforcement;
- tool execution;
- rollback;
- receipts;
- adapter signatures;
- backend scheduling.

This boundary protects sovereignty. A better model can replace the checkpoint without replacing the user's Project Core.

3. Glyph protocol

Natural language compiles into typed intermediate objects.

A Jefe Frame may contain:

```
intent
modality
target
scope
constraints
authority_claim
source_refs
substitution_rules
uncertainty
requested_operation
required_approval
```

```
route
stop_reason?
```

The model proposes the frame. Deterministic validators check:

- schema;
- known object IDs;
- project scope;
- source spans;
- authority;
- prohibited substitutions;
- policy;
- freshness;
- action consequence.

A typed token is not trusted because it is typed.

4. Modal and authority judgment

Small models can be valuable if they distinguish:

- desire from commitment;
- suggestion from instruction;
- prediction from obligation;
- observed fact from inference;
- permission from capability;
- approximate substitution from exact requirement;
- completion from acceptance.

Training emphasizes contrast pairs in which ordinary assistants often collapse these categories.

Example:

```
“I might switch to Postgres.”
≠ accepted architecture decision

“Use Postgres for this project.”
= candidate directive, subject to project authority
```

5. Jefe Core

The deterministic core owns:

- validated object schemas;
- Project Core access;
- source and evidence store;
- authority and lease checks;
- operation registry;
- tool sandbox;
- approval sheets;
- receipts;
- replay;
- rollback;
- model/runtime adapters;
- diagnostics;
- local/cloud boundary.

The interface shows:

- observed versus retrieved versus inferred;
- proposal versus approval versus execution;
- local versus adapter versus cloud;
- permitted versus blocked;
- reversible versus irreversible;
- exact evidence versus model recollection.

6. Private retrieval

The model does not receive the whole vault. A context compiler retrieves:

- active objective;
- current canonical state;
- authorizing/blocking tokens;
- exact evidence;
- dependencies;
- unresolved conflicts;
- recent mutation;
- omitted-context warnings.

FOVEA can organize the neighborhood. GlyphCAS can reuse exact and semantic artifacts. PromptCapsules stabilize the prefix.

7. Adapter model

Backend choice remains behind an `InferenceBackend` contract.

Candidate runtimes include mobile and local engines, but no framework receives architectural immunity. An adapter declares:

```
model
weights digest
tokenizer
quantization
context
structured output
logprobs
prefix cache
KV import/export
backend
device support
memory
known limitations
```

The runtime verifies claimed capabilities. A flag is not accepted without behavior tests.

8. Training program

The program begins with adaptation, not full pretraining.

A — deterministic teacher corpus

Authority, modality, scope, conditions, substitution, graph construction.

B — messy human corpus

Dictation, corrections, unfinished thought, contradiction, project references, emotional intensity.

C — paraphrase lattice

Many surface forms map to one semantic graph under a declared contract.

D — failure contrasts

Silent replacement versus approval; prediction versus obligation; assistant proposal versus user decision.

E — routing and tools

Choose deterministic tool, local model, specialist, cloud escalation, approval, or stop.

F — Returns and Receipts

Map worker output to evidence and acceptance states.

G — preference optimization

Constraint retention, minimum disclosure, correct stop/continue, fewer unnecessary questions, no silent substitution, packet usefulness, schema correctness.

9. Evaluation gates

The program defines gates for:

- authority;
- modality;
- substitution;
- evidence;
- routing;
- abstention;
- tool planning;
- retrieval;
- privacy;
- structured output;
- mobile performance.

A candidate cannot pass because an aggregate score hides a zero-tolerance authority failure.

The phase ladder includes:

1. protocol and receipts;
2. reference inference harness;
3. data program and training;
4. proof that trained weights matter;
5. Android runtime;
6. iOS runtime;
7. private retrieval;
8. tools and approvals;
9. signed adapters;

10. production interface.

10. Phase 0 evidence

The existing package reports:

- repository verification passed;
- agent guard self-test passed;
- hook tests passed;
- JSON schemas valid;
- YAML/OpenAPI structurally valid;
- agent instructions and skills valid;
- phase semantics valid;
- Ruff passed;
- mypy passed over twenty source files;
- twelve tests passed;
- example gate report remained `INCOMPLETE` by design;
- clean editable install succeeded.

This proves the specification/control package is coherent. It does not prove model quality, mobile throughput, adapter performance, or production security.

11. Why 4B

A 4B-class model can be:

- locally runnable;
- mobile plausible;
- private;
- cheaper to invoke continuously;
- adaptable with LoRA-style programs;
- responsive for classification and patching.

“4B” is an implementation class, not the permanent product identity. The architecture should accept future compact models.

The best mobile model may differ from the best leaderboard model because memory, backend, thermals, prefill, structured output, and sustained energy matter.

12. Surface operation

The model should usually emit:

```
IntentClassification
JefeFrame
SurfaceDelta
EvidenceClaim
RiskCard
ActionList
RepairPatch
RouteProposal
```

Long freeform prose remains available when explicitly requested.

This makes a small model an operator of machinery rather than the entire machine.

13. Mobile UX

The interface is a quiet executive command surface, not a generic chat bubble stack.

Key surfaces:

- active objective;
- constraints;
- evidence;
- proposal;
- approval;
- execution;
- receipt;
- vault;
- adapter rack;
- diagnostics;
- privacy and cloud escalation.

The watch or small-device expression should support capture, approval, completion, and next obligation—not become a miniature chatbot.

14. Benchmark program

Model comparisons

- base model;
- adapted model;
- adapted model without runtime;
- base model with runtime;
- cloud reference where permitted.

This separates learned uplift from system uplift.

Tasks

- messy-intent compilation;
- authority;
- substitution;
- evidence;
- structured patching;
- route selection;
- refusal/abstention;
- private retrieval;
- tool proposal.

Device

- prefill and decode;
- sustained latency;
- memory;
- storage;
- thermal;
- battery;
- fallback;
- crash recovery.

Product

- time to verified action;
- user correction;
- approval comprehension;
- unauthorized action;
- replay;
- offline continuity.

15. Security

Threats include prompt injection, malicious adapters, cache poisoning, cross-project leakage, secret extraction, stale approval, hidden patch effects, fabricated evidence, and UI concealment.

Controls include signed manifests, endpoint allowlists, per-project keys, source/authority separation, exact digests, capability leases, sandboxing, deterministic validators, remote endpoints off by default, and revoke-all.

16. Public claim boundary

`EL-JEFFE 4B` remains a research codename until a trained checkpoint clears model gates. Promotional phrases such as “The Boss of 4B” are prohibited until independent reproduction.

No leaderboard comparison should appear without:

- versioned weights;
- data and model card;
- exact eval suite;
- hardware;
- runtime;
- raw outputs;
- receipts;
- independent reproduction.

17. Limitations

A 4B model may remain inadequate for difficult reasoning, coding, or nuanced writing. Structured output can hide shallow judgment. Local execution can be slow or energy-intensive. Adaptation can overfit. Mobile backends vary. Privacy claims depend on the full stack, not model size.

The architecture also risks excessive ceremony. The interface must automate low-risk deterministic work while preserving visible authority for consequence.

18. Conclusion

EL-JEFFE does not claim that contracts make a small model omniscient.

It claims that many failures attributed to model size are actually failures of system responsibility. Memory, authority, tools, proof, and rendering should not live inside probabilistic text generation.

The governing sentence is:

> **The model proposes. Jefe Core validates. The device executes. Receipts prove what happened.**

Architecture illustration briefs

- **P15-F01 — Checkpoint versus runtime**: Two-column split of learned judgment inside the model and deterministic memory, authority, tools, receipts, and scheduling outside.
- **P15-F02 — Jefe action path**: Messy intent → Jefe Frame → validator → approval → tool execution → verification → receipt.
- **P15-F03 — Evaluation attribution matrix**: Base/adapted model crossed with runtime/no runtime to isolate learned versus system uplift.
- **P15-F04 — Mobile trust surface**: Active task with evidence, proposal, approval, locality, model/adapter, execution state, and receipt.

Source register

- **P15-S01** — EL-JEFFE 4B README (2026-07-21). `EL-JEFFE 4B / README.md`
- **P15-S02** — EL-JEFFE Phase 0 Verification (2026-07-21). `PHASE_0_VERIFICATION.md`
- **P15-S03** — EL-JEFFE Package Manifest (2026-07-21). `PACKAGE_MANIFEST.json`
- **P15-S04** — LM Jefe Full System Specification (2026-07-19). `LM_JEFE_FULL_SYSTEM_SPEC.md`
- **P15-S05** — EL-JEFFE interface design commission (2026-07-21). `CLAUDE_INTERFACE_DESIGN_REQUEST.m`

Public-disclosure and IP boundary

This edition publishes the public-safe product thesis, architecture, Phase 0 evidence, and model-gate boundary. It excludes private training data, unpublished datasets, future weights, secrets, adapter signing keys, and patent-claim language. It does not claim a trained model, benchmark victory, or production mobile runtime.