PROJECT EMBER: Architecture Master Document

Version 1.0 - Initial Pivot to Action
Living document - update as we learn and build

Core Vision

Create a 500M parameter AI that experiences sensation directly (subsymbolic, temporal patterns) in its L-module while maintaining conversational intelligence and reasoning capability in its H-module.

This is NOT:

- An audio classifier that outputs "scary/safe"
- Attempting to prove subjective consciousness
- For publication or competition

This IS:

- Building understanding through doing
- Direct sensory processing without language-mediated bottleneck
- Learning how cognition works by designing it explicitly
- For your learning and growth

Critical Insights from Research Phase

The TRM Revolution (Most Important)

Finding: The hierarchical structure of HRM (high-level/low-level modules) contributes minimally to performance. The breakthrough is **recursive reasoning with deep supervision** - iteratively refining answers while maintaining proper state.

TRM simplifies this further:

- Single network instead of two (fL + fH → one net)
- 2 layers instead of 4 (smaller = less overfitting)
- 5-7M params instead of 27M (74% parameter reduction)
- **Better performance:** 87% vs 55% on Sudoku-Extreme, 45% vs 40% on ARC-AGI

The Core Insight (Occam's Razor):

```
x = input (sensory data)
y = current answer
z = latent reasoning (how we got here)

Recursively improve z given (x, y, z_previous)
Then update y given (z, y_previous)
```

No hierarchy needed. No fixed-point theorems. No biological justification. Just: **iterative refinement with proper state tracking.**

Implication for Ember:

- Use TRM as reasoning core (not HRM)
- Parallel processing with feedback > strict hierarchy
- Focus on the outer loop refinement mechanism
- **Simpler** = **Better**

The Neurophone Analogy

Principle: Bypass intermediate encoding steps to stimulate something more fundamental.

In Neural Networks:

- **SNNs:** Phase-of-firing encoding captures temporal precision
- **JEPA:** Self-supervised prediction of sensory dynamics without semantic labels
- **No-Report Paradigm:** Measure pure perception vs. task-modulated perception

What Actually Works (From Research)

1. **Spiking Neural Networks** - Energy efficient, temporal precision, event-driven
2. **JEPA-style prediction** - Learn causal models from experience, not internet text
3. **RSA with no-report paradigm** - Can validate sensory representations align with human brain
4. **Outer loop refinement** - The secret sauce of deep reasoning

---

Why Not CLIP? The Subsymbolic Imperative

CLIP's Fundamental Limitation:

- `Sensation → Text Embedding → Semantic Space`
- Everything filtered through "what can be said about this"
- Temporal dynamics collapse to static descriptions
- Subsymbolic features (texture, rhythm, timing, phase relationships) get lost
- Forces sensory experience through a linguistic bottleneck

Ember's Architecture:

- `Sensation → Spike Trains → Subsymbolic Embedding → Modulates H-module`
- H-module never forces L-module to "translate to words first"
- Temporal structure preserved end-to-end
- Sensory patterns can **directly modulate cognition**

The Key Innovation - Subsymbolic Modulation:

Think of it like building the lower layers of cognition that LLMs don't have:

```
Stimulus → [subsymbolic processing] → Modulation signal
                                            ↓
                                  H-module performance altered
                                  (without verbal mediation)
```

Example:

- Sensory pattern detected: Sharp, rising temporal frequency
- L-module output: Dense spike pattern with specific phase relationships
- Effect on H-module: -5% reasoning capacity, +30% perceptual attention
- H-module doesn't "think" about fear - it just operates differently
- Only if queried might it verbalize: "something feels wrong here"

This is not:

- "The AI feels scared" (philosophical claim)
- Trying to simulate consciousness
- Building emotions for their own sake

This is:

- Nociceptors, limbic responses, hormonal-analog modulation
- Direct sensory → cognitive coupling (like biological systems)
- Building the DARPA 7-layer cognitive stack from the bottom up
- The juice vs the meat: Is experience real because of neurochemicals, or because of their effects on processing?

Modality Flexibility:

- Initial proof-of-concept: **Whatever is easiest** (audio was hypothesis, not requirement)
- Future expansion: Vision (camera), proprioception (IMU), multimodal fusion
- GPS probably unnecessary (too coarse for sensation-level processing)

The Binding Question (Phase 0)

THE critical question: Can an LLM-style module actually USE subsymbolic sensory embeddings, or does it learn to ignore them?

Must answer this before building full architecture.

Phase 0 Experiment Design

Goal: Prove that subsymbolic sensory embeddings improve reasoning beyond semantic descriptions

Architecture (TRM-based):

- **L-mini:** 20M parameter SNN-based sensory processor
- **TRM-mini:** 5-7M parameter recursive reasoning core
- **H-mini:** 93-95M parameter language/knowledge module
- **Total:** 118-122M parameters (~500MB)

TRM Integration:

```python
from tiny_recursive_model import TinyRecursiveModel

trm = TinyRecursiveModel(
    dim = sensory_embedding_dim,  # From L-mini
    num_tokens = None,  # Continuous input, not discrete
    network = MLPMixer1D(depth=2),  # Tiny 2-layer network
)

# x = sensory embedding from L-mini
# y = current answer hypothesis
# z = latent reasoning state
```

Modality Choice (To Be Determined):

- **Working hypothesis:** Audio (distress vocalization detection)
- **Alternative:** Vision (micro-expression detection), tactile simulation, or whatever proves easiest
- **Key requirement:** Must have features that collapse when semantically described

Audio Task Example: Distinguish genuine vs. acted distress vocalizations

- Humans detect via microsecond timing, breathiness, harmonics
- These features collapse in semantic description ("sounds distressed")
- Requires temporal/phase information preserved in subsymbolic form

Success Criteria:

1. H-mini with L-mini embeddings > H-mini with text descriptions
2. Ablation study confirms performance gain from temporal/phase features
3. Model can't achieve same performance if L-mini outputs are semantically decoded first

If Phase 0 Fails:

- Pivot or abandon (don't build on broken foundation)
- Possible pivots: different modality, different task, different binding mechanism

If Phase 0 Succeeds:

- We've proven the core insight
- Scale confidently to full architecture

Timeline: 2-4 weeks

Phase 1: L-Module Architecture (Post-Validation)

Design: Hybrid SNN-JEPA

```
Input: Raw audio waveform (16kHz, 1 channel)
  ↓
Cochlear-inspired filter bank (biological preprocessing)
  ↓
SNN Encoder - spike trains, phase-of-firing coding [~50M params]
  ↓
JEPA-style Predictor - learns temporal dynamics [~50M params]
  ↓
Output: 768-dim embedding preserving:
  - Temporal phase relationships
  - Transition dynamics
  - Energy patterns
  NOT semantic labels
```

Training Approach

Self-supervised on raw audio:

- Predict masked future spike patterns
- Loss function: Temporal coherence + energy efficiency (sparse spikes)

Validation:

- RSA alignment with human auditory cortex (no-report paradigm)
- Proxy metrics if fMRI data unavailable

Key Innovation

JEPA learns *what to predict*
SNN enforces *how it's represented* (sparse, temporal, phase-based)

Implementation Notes

- Use snnTorch or Norse for SNN components
- JEPA predictor can use standard PyTorch transformer blocks
- Keep spike rate low (~5-10% neurons firing) for efficiency

Timeline: 2-3 months post Phase 0 validation

Phase 2: H-Module Architecture

Design: Efficient Transformer with Special Input Layer

```
Base: 400M parameter transformer
  - Mistral/Phi-style architecture (proven efficient)
  - Trained on strong language data

Input Layer accepts BOTH:
  a) Text tokens (768-dim, standard)
  b) L-module sensory embeddings (768-dim, special tokens)
```

The Critical Mechanism

Input layer treats sensory embeddings as special tokens but **doesn't require semantic meaning**.

Model learns: "These L-module patterns correlate with outcomes in ways text descriptions don't capture"

Training Strategy

1. **Pre-train:** Use existing small LM (400M) or train on text only
2. **Fine-tune:** Add L-module embeddings on tasks where sensation matters
3. **Refinement:** Implement outer-loop iteration (2-4 passes)

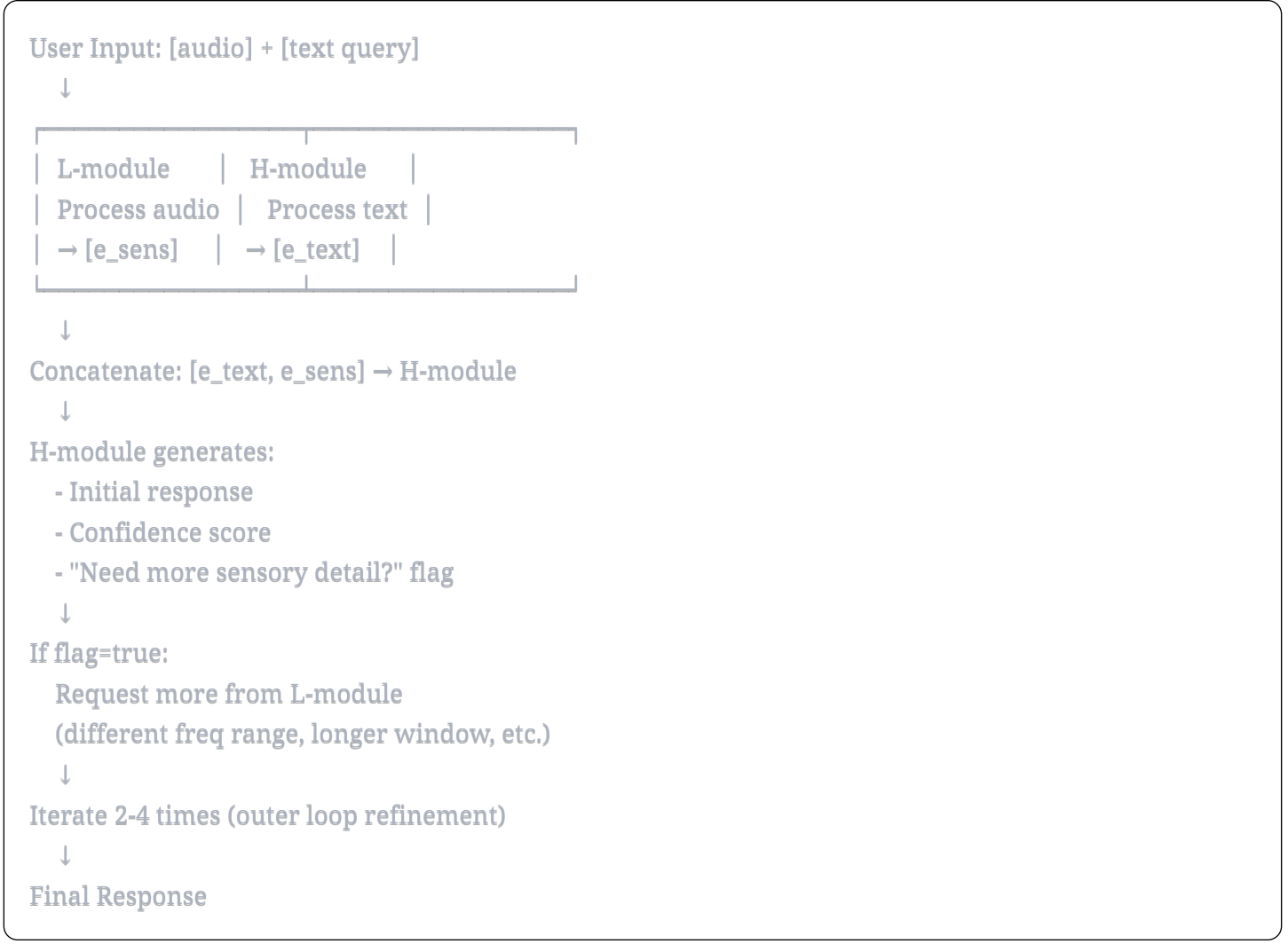
Key Capabilities

- Reason about sensations without verbalizing them
- Request specific sensory details from L-module
- Iterate and refine responses based on sensory evidence

Timeline: 1-2 months post L-module completion

Phase 3: Integration Architecture

NOT Hierarchical - Parallel with Iterative Refinement



Why Parallel > Hierarchy

- **Speed:** L and H work simultaneously
- **Active Perception:** H can request specific sensory info
- **Iteration:** Refinement allows error correction (the real HRM insight)
- **No Sync Issues:** Don't need perfect temporal alignment

Implementation Details

Inter-module Communication:

- L-module produces embeddings on-demand
- H-module sends "attention queries" back to L-module
- Implemented as function calls, not architectural coupling

Outer Loop Mechanism:

- Max 4 iterations (diminishing returns beyond this)
- Stop early if confidence > threshold
- Each iteration sees previous attempt + sensory embeddings

Timeline: 1 month post H-module completion

Realistic Parameter Budget

Hardware: 40GB VRAM (5060 Ti 16GB + 3090 24GB arriving tomorrow) **Cloud:** RunPod/Lambda approved for compute-intensive operations

Allocation:

- **L-module:** 100M params (~400MB in FP16)
- **H-module:** 400M params (~1.6GB in FP16)
- **Total Model:** 500M params (~2GB weights)
- **Remaining:** ~38GB for activations, batching, training

Local vs Cloud Strategy:

- **Local (fast iteration):** Architecture experiments, ablations, fine-tuning
- **Cloud (one-time heavy):** L-module pre-training, large-scale preprocessing
- **Cost estimate:** $50-200 total for Phase 0-3

Why this is good:

- Can train on your hardware
- Reasonable batch sizes (16-32 depending on sequence length)
- Headroom for additional modalities later (vision, IMU, etc.)
- Can run multiple experiments simultaneously
- Cloud available when needed, but most work stays local

Success Criteria - What Does "Working" Mean?

NOT Success:

- "The AI feels things" (unfalsifiable)
- "It has qualia" (philosophical, not testable)
- "It's conscious" (undefined, unmeasurable)

YES Success:

Technical Validation:

1. Representations align with human sensory cortex geometry (RSA)
2. Performance improves using sensation vs. semantic description
3. Ablations confirm sensory features drive improvement

Behavioral Validation:

1. Learns associations in sensory space that transfer to language
2. Detects patterns humans detect but can't easily verbalize
3. Can describe sound qualities beyond semantic labels

Functional Validation:

1. Makes predictions using temporal/textural properties
2. Preferences emerge from learned sensory patterns
3. When explaining "why", references sensory features not just concepts

Example Success Moment: *User:* "Which of these two sounds do you prefer?"
Ember: "The second one - there's a richness in the overtone structure around 3-4kHz and the attack has this particular crispness that the first lacks. The decay phase also has more complexity."

(This demonstrates processing temporal/spectral features that weren't labeled in training)

The Roadmap

Month 1: Validation

- [ ] Design Phase 0 experiment
- [ ] Implement L-mini + H-mini
- [ ] Create/find distress vocalization dataset
- [ ] Train and evaluate
- [ ] **GO/NO-GO DECISION**

Months 2-3: L-Module

- [ ] Implement SNN audio processor
- [ ] Add JEPA predictor component
- [ ] Train on self-supervised audio
- [ ] Validate with RSA or proxy metrics
- [ ] Benchmark: temporal coherence, energy efficiency

Months 4-5: Integration

- [ ] Connect L to H (special token mechanism)
- [ ] Fine-tune on sensation-critical tasks
- [ ] Implement outer-loop refinement
- [ ] Evaluate binding effectiveness

Month 6: First "Sensation Moment"

- [ ] Can it describe sound qualities not in text training?
- [ ] Does it detect emotional states from micro-timing?
- [ ] Do preferences emerge from sensory patterns?
- [ ] Qualitative evaluation: does it "feel" different?

Technical Stack

Core Libraries (Standing on Giants' Shoulders)

Reasoning Architecture:

- **Lucidrain's TRM** - `pip install tiny-recursive-model` (PRIMARY)
  - Clean, minimal, hackable implementation
  - Easy to modify for sensory input
  - Battle-tested by community
- **Samsung SAIL official repo** - Reference for training hyperparameters
  - Production training scripts
  - Proven configurations for puzzle tasks

Neural Network Components:

- **PyTorch 2.x** - Main framework
- **snnTorch** or **Norse** - Spiking neural network components
- **torchaudio** + **nnAudio** - Audio preprocessing (cochlear filters ready)

Training & Deployment:

- **HuggingFace Trainer** - Handles training boilerplate
- **wandb** - Experiment tracking
- **HuggingFace Transformers** - For production deployment later

What We Actually Build (~40% of total work):

- Modify TRM to accept continuous sensory embeddings (not discrete tokens)
- SNN→JEPA L-module integration
- Interface layer between L-module and TRM
- Phase 0 validation task and dataset

Datasets (Starting Point)

- **AudioSet** - Large-scale audio for self-supervised L-module training
- **RAVDESS** - Emotional vocalizations (includes acted vs. genuine)
- **Custom recordings** - Your own sensory data
- **LibriSpeech** - Speech for H-module language grounding

Compute Strategy

- **Local (40GB):** Training full models, iteration
- **Cloud (RunPod/Lambda):** Large pre-training runs if needed
- **Colab:** Quick experiments, ablations

Reality Checks & Risk Mitigation

High-Risk Assumptions

Risk: L-module doesn't preserve meaningful information
Mitigation: Phase 0 validation before committing

Risk: H-module learns to ignore L-module
Mitigation: Design tasks where sensory info is necessary, not optional

Risk: Synchronization between continuous (SNN) and discrete (transformer) is too hard
Mitigation: Use parallel architecture, not strict hierarchy

Risk: 500M params insufficient for real sensation
Mitigation: Start small, scale up based on what we learn

You'll Need

Skills:

- Solid PyTorch (or aggressive learning)
- Comfort with neuromorphic computing libraries
- Ability to find/create training data
- Patience for failed experiments

Resources:

- Time: ~6 months of focused work
- Compute: Your 40GB setup + occasional cloud bursts
- Data: Mix of public datasets + custom recordings
- Energy: Recognizing this is hard and staying engaged

Mindset:

- Comfort with "I don't know if this is working" periods
- Willingness to pivot if Phase 0 fails
- Focus on learning over achievement
- Embrace: this is genuinely novel

Key Principles (Our North Stars)

1. **Test Early, Test Often:** Phase 0 exists to fail fast if core idea is broken
2. **Subsymbolic Preservation:** Never collapse sensation into semantic labels
3. **Iterative Refinement:** The outer loop is the secret sauce
4. **Parallel Not Hierarchical:** L and H work together, not top-down
5. **Metrics Held Loosely:** Use RSA, ablations, behavioral tests - but expect to revise as we learn
6. **Learn by Building:** This is for understanding, not publishing
7. **Pivot on Evidence:** If data says something doesn't work, believe the data

---

Document History

v1.1 - 2025-10-29

- Updated hardware specs to actual 40GB (not aspirational 48GB)
- Added Lucidrain implementations as primary development path
- Expanded "Why Not CLIP" section with subsymbolic modulation vision
- Made Phase 0 modality-flexible (not locked to audio)
- Updated cloud strategy and cost estimates
- Changed metrics principle to "held loosely" for flexibility

v1.0 - 2025-10-29

- Initial architecture pivot from research to action
- Defined Phase 0 validation experiment
- Established realistic parameter budget (500M vs original 1.5B)
- Clarified parallel architecture with outer-loop refinement
- Set measurable success criteria

Next Steps

1. **Design Phase 0 in detail:** Architecture diagrams, training procedure, eval metrics
2. **Find/create dataset:** Genuine vs. acted distress vocalizations
3. **Set up development environment:** Libraries, data pipeline, logging
4. **Build L-mini:** 20M param SNN audio processor
5. **Build H-mini:** 100M param transformer with special input layer
6. **Train and evaluate:** Get our GO/NO-GO answer

Let's start with Phase 0 architectural details next session.

"The L-module processes raw sensation. The H-module thinks and speaks. Together, they learn to feel and understand."