

Pragnosia v0.1

Vision

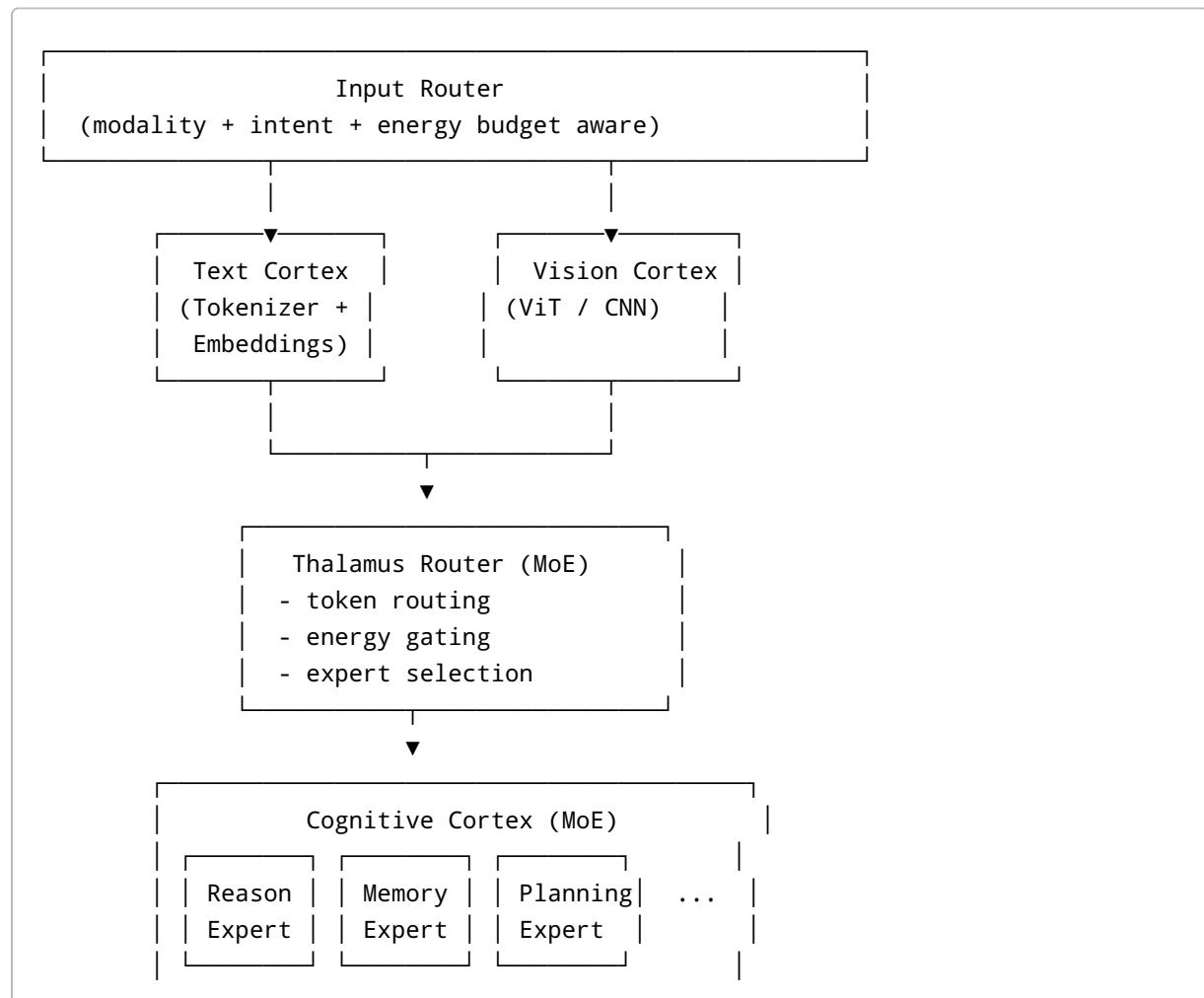
Pragnosia is a brain-inspired, energy-efficient, modular, multimodal large language model designed to **run, fine-tune, and infer on consumer-grade hardware (4GB GPU, 16-24GB RAM)** while **scaling architecturally from 1B to 450B parameters** through sparsity, routing, and plasticity.

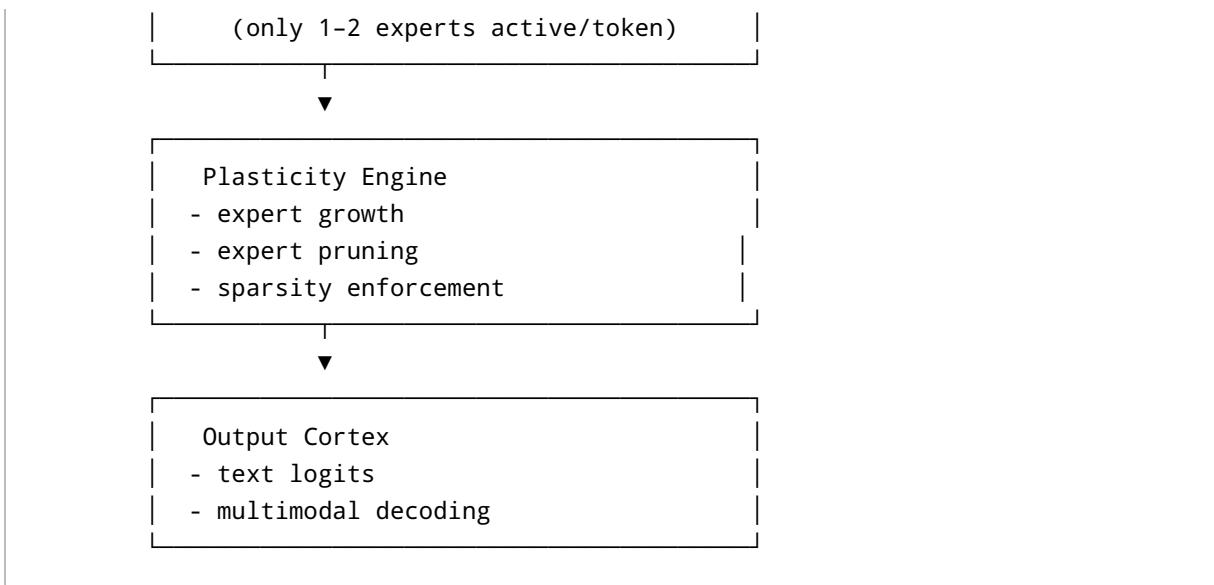
The core principle is:

Total parameters are large, but active parameters per forward pass are small.

1. System Architecture (Pragnosia v0.1)

High-Level Pipeline





Brain Analog Mapping

Brain Concept	Pragnosia Module
Thalamus	Token / Expert Router
Cortex regions	MoE Experts
Neuroplasticity	Dynamic expert growth/prune
Attention	Sparse routing
Metabolic budget	Energy-aware gating
Memory	KV cache + memory expert

2. Exact Module Interfaces

2.1 Input Router

Responsibility - Detect modality - Estimate complexity - Assign energy budget

```

class InputRouter:
    def forward(
        self,
        raw_input: Any,
        modality: Literal["text", "image", "audio"],
        max_energy: float
    ) -> RoutedInput

```

Output:

```
RoutedInput = {
    "tokens": Tensor[B, T, D],
    "modality": str,
    "energy_budget": float
}
```

2.2 Vision Cortex (Multimodal)

- Frozen encoder (CLIP-ViT / MobileViT)
- Projection to LLM embedding space

```
class VisionCortex(nn.Module):
    def forward(self, image: Tensor[B, C, H, W]) -> Tensor[B, N, D]
```

2.3 Thalamus Router (Core Innovation)

Responsibilities - Token-wise expert routing - Top-K sparse activation - Energy-aware gating

```
class ThalamusRouter(nn.Module):
    def forward(
        self,
        tokens: Tensor[B, T, D],
        energy_budget: float
    ) -> Dict[ExpertID, Tensor]
```

Only selected experts execute; others remain inactive.

2.4 Cognitive Cortex (Experts)

Each expert is independent and modular.

```
class Expert(nn.Module):
    def forward(self, tokens: Tensor[B, T, D]) -> Tensor[B, T, D]
```

Expert types in v0.1

Expert	Function
Language	Syntax & fluency
Reasoning	Math & logic
Memory	Long-context recall
Planning	Step-by-step reasoning

2.5 Plasticity Engine

v0.1 scope: expert-level plasticity (not per-neuron yet)

```
class PlasticityEngine:
    def grow_expert(self, reason: str) -> None
    def prune_expert(self, expert_id: int) -> None
```

Growth triggers - High routing entropy - Persistent loss spikes - Expert saturation

Pruning triggers - Low activation frequency - Redundant gradients

2.6 Output Cortex

```
class OutputCortex(nn.Module):
    def forward(self, tokens: Tensor[B, T, D]) -> Tensor[B, T, V]
```

3. 30-Day Execution Plan

Week 1 – Core Skeleton

Goal: runnable text-only Pragnosia

- Repo setup, config system
- Tokenizer + base Transformer
- MoE layer + Thalamus router
- CPU/GPU offloading
- Gradient checkpointing
- 4-bit weight loading

Deliverable: - ~1B sparse MoE model - Inference on 4GB GPU

Week 2 – Plasticity & Energy Awareness

- Expert activation tracking
- Energy budget simulation
- Hard gating enforcement
- Expert growth logic
- Expert pruning logic

Deliverable: - Dynamic expert creation/deletion - Active params \ll total params

Week 3 – Multimodal & Scaling

- Vision encoder integration
- Image-text alignment
- Scale to 3-7B total params
- LoRA-only training
- Memory expert for long context

Deliverable: - Multimodal Pragnosia v0.1

Week 4 – Data, Training & Evaluation

- Dataset preprocessing
- Curriculum training
- Fine-tuning
- Benchmarking vs GPT-OSS-20B tasks
- Quantized release
- Documentation

Deliverable: - Open-source Pragnosia v0.1 - Reproducible 4GB pipeline

4. Data Strategy (GPT-OSS-20B Competitive)

Text Data

Source	Purpose
FineWeb-Edu	High-quality language
OpenWebMath	Reasoning
Code datasets	Logic & structure
Wikipedia + Books	Knowledge

Multimodal Data

Source	Purpose
LAION-Aesthetics	Vision grounding
COCO	Image-caption alignment
OCR datasets	Text-vision fusion

Curriculum Order

1. Language-only stabilization
2. Reasoning-heavy text
3. Multimodal alignment
4. Instruction tuning

5. Expected Outcomes

Dimension	Result
Raw fluency	Slightly below GPT-OSS-20B
Reasoning	Competitive
Efficiency	Orders of magnitude better
Hardware accessibility	4GB-first
Research novelty	Very high

Closing Notes

Pragnosia v0.1 is not an attempt to outscale frontier models. It is an attempt to **redefine efficiency, adaptability, and accessibility** by designing an LLM that behaves more like a brain than a static matrix.

This document defines a concrete, implementable starting point.