NEURO-NANO

Master Operational Manual

Deployment & Tactical Guide

Target Hardware: CPU (x86-64 / ARM64)
Environment: Offline / Air-Gapped

Operations Division

January 27, 2026

Contents

1 Phase 1: Installation & Deployment

This guide assumes a standard consumer laptop (Windows, Mac, or Linux) and an intent to run **100% locally** without internet dependency. We utilize the `llama.cpp` ecosystem for high-efficiency CPU inference.

1.1 Option A: The GUI Method (LM Studio)

Best for: Rapid usage, visual interface, and beginners.

1. **Install the Interface:**
   - Download **LM Studio** from `lmstudio.ai`.
   - Run the installer. Requires $\approx$ 400MB space.
2. **Acquire the Model Artifact:**
   - If you have not forged your own model, search for: `Qwen2.5-1.5B-Instruct-GGUF`.
   - **Critical:** Select the **Q4_K_M** quantization.
   - *Why:* This is the "Golden Ratio" of speed vs. intelligence ($\approx$ 980MB).
3. **Mount the Model:**
   - Open the "AI Chat" tab.
   - Select the downloaded model from the top dropdown.
   - **Verify:** Ensure the RAM meter at the top shows only $\approx$ 1GB usage.

1.2 Option B: The Terminal Method (Command Line)

Best for: Automation, lowest latency, and scripting.

1. **Deploy the Engine:**
   - **Windows:** Download the latest `llama-bin-win-avx2-x64.zip` from GitHub. Extract to a dedicated folder (e.g., `C:\neuro-nano`).
   - **Mac/Linux:** Install via Brew: `brew install llama.cpp` or compile from source.
2. **Deploy the Asset:**
   - Place your `nano.gguf` file in the same directory as the `llama-cli` executable.

2 Phase 2: Operational Tactics

2.1 System Configuration (The "Soul")

To activate the specific reasoning capabilities of NEURO-NANO, you must inject the correct System Prompt. Do not use the default "Helpful Assistant" persona.

```
"You are NEURO-NANO, a high-density logical reasoning engine. You answer
    succinctly, accurately, and without filler. You prioritize code
   correctness and logical steps. You are offline and private."
```

Listing 1: System Prompt Injection

2.2 Inference Parameters (The "Physics")

Configure your runtime settings to match the model's small architecture:

- **Context Length:** `8192` (Safe maximum for speed).
- **Temperature:** `0.6` (Low temp reduces hallucination in logic tasks).
- **Repeat Penalty:** `1.1` (Prevents "loops" common in small models).
- **CPU Threads:** Set equal to your **physical** core count (not logical).

2.3 Command Line Operations

1. Interactive Chat Mode:

```
./llama-cli -m nano.gguf -n -1 --color -cnv -p "You are NEURO-NANO."
```

- `-n -1`: Infinite text generation.
- `-cnv`: Conversation mode (keeps history).
- `-color`: Visual separation of User/AI.

2. One-Shot Task Mode (Automation):

```
./llama-cli -m nano.gguf -p "Write a Python script to scan for .txt
   files." --no-display-prompt
```

3 Phase 3: Prompt Engineering Protocols

Small Language Models (SLMs) require **Command Precision**. They lack the massive world-knowledge of GPT-4, so they must be guided.

3.1 Protocol 1: Chain of Thought (CoT)

Always force the model to show its work. This triggers the logic circuits trained on the Orca Math dataset.

- **Wrong:** "Solve: $24 \times 15 + 12$."
- **Correct:** "Solve: $24 \times 15 + 12$. **Think step-by-step.**"

3.2 Protocol 2: The Coding Constraint

Prevent the model from wasting tokens on conversational filler ("Here is your code", "I hope this helps").

- **Command:** "Write a C++ sorting function. **Provide code only. Do not explain.**"

3.3 Protocol 3: Document Ingestion

- **Capacity:** The model supports 32k context, but reasoning degrades after 16k.
- **Limit:** Keep input documents under $\approx$ 20 pages for maximum accuracy.
- **CLI Usage:** Use the `-f` flag to load a file:

```
./llama-cli -m nano.gguf -f secret_memo.txt -p "Summarize this
."
```

4 Phase 4: Troubleshooting

Symptom	Remediation
Loop of Death	The model repeats the same sentence infinitely. **Fix:** Increase `Repeat Penalty` to 1.2.
Gibberish Output	Model outputs symbols or alien text. **Fix:** Temperature is too high. Lower to 0.1.
Slow Speed (<10 t/s)	CPU bottleneck. **Fix:** Check thread count. Ensure you are not using "Hyper-threading" cores, only physical ones.
Hallucination	Inventing facts. **Fix:** The model is a logic engine, not a search engine. Provide the facts in the prompt (RAG).