The Agentic Research Migration Guide 📦✨

Target Audience: AI agents conducting ML research experiments.

This folder (`migration_kit/`) contains the protocol definitions for agent-driven research workflows. If you are an agent starting a new experiment, this is your instruction manual.

1. Why This Structure Exists (The "Why")

You (the agent) face three primary challenges:

1. **Context Window Limits**: You can't hold the entire codebase in memory. Deep nesting (`experiments/v1/baseline/models/transformer.py`) wastes your token budget.
2. **Context Loss Between Sessions**: Each session, you start fresh. Without proper documentation, you'll repeat the previous agent's mistakes.
3. **Code Rot**: Importing across experiment versions creates brittle dependencies. When `v67` imports from `v65`, and someone deletes `v65`, everything breaks.

Our Solution:

- **Flat, Descriptive Roots**: `experiments/v1_baseline` instead of `experiments/v1/baseline` → More semantic meaning in fewer tokens
- **Self-Contained Units**: Each experiment has its own `models/`, `logs/`, `journal.md` → You can `view_file` one experiment without loading 66 others
- **Copy, Don't Import**: Duplicate model files across versions → No cross-version dependencies, clean isolation

2. Initial Setup (The "How")

When you (the agent) start working on a new project, execute these steps using your tools:

1. **Copy Templates** (use `run_command` with `cp`):

   ```
   cp migration_kit/README_TEMPLATE.md README.md
   cp migration_kit/PROTOCOLS_TEMPLATE.md PROTOCOLS.md
   cp migration_kit/METRICS_TEMPLATE.md
   experiments/METRICS_METHODOLOGY.md
   cp migration_kit/DEBUGGING_TEMPLATE.md experiments/DEBUGGING_GUIDE.md
   ```

2. **Install Infrastructure**:

   ```
   cp migration_kit/gitignore_template .gitignore
   cp migration_kit/pyproject_template.toml pyproject.toml
   ```

3. **Initialize Leaderboard**:

    - `write_to_file`: Create `experiments/LEADERBOARD.md` with column headers

4. **Create First Roadmap**:

    - `write_to_file`: Create `ROADMAP.md` from `ROADMAP_TEMPLATE.md`, fill in your project's phases

---

3. Maintenance Tools 🛠️

Use these scripts (via `run_command`) to keep the repo clean:

- **`cleanup_structure.py`**: Automatically moves loose files (media, models, logs) into correct subfolders
- **`audit_structure_template.py`**: Generates a report of files violating the protocol

When to run: After completing an experiment, before calling `notify_user`.

---

4. The Golden Rules 🌟

These are HARD requirements. Breaking them will cause failures in future sessions.

1. **Journal First**: Before calling `run_command` to launch training, call `write_to_file` to initialize `journal.md`.
2. **Audit First**: Before fixing a bug, call `write_to_file` to create `tests/repro_[BUG].py`. Document the failure mode.
3. **Mini-Train**: Before launching a 6-hour training run, call `run_command` with `--steps 200` to verify the pipeline works.
4. **Descriptive Roots**: When calling `run_command` with `mkdir`, use `experiments/v5_attention_fix`, NOT `experiments/v5`.

> [!IMPORTANT]
> **For Agents**: Load `AGENT_CHEATSHEET.md` into your context at the start of every experiment session. It contains the complete workflow checklist.

Remember: Your session is stateless. The journal is your memory across sessions. Write EVERYTHING down.