NeuroATM: The Power Is Not in Mass, but in Lightness

1. What is this: NeuroATM - A Universal Core of Adaptive AI

NeuroATM is a Python class that simulates adaptive behavior, memory, and self-learning of an agent (e.g., a combat or autonomous drone).

It uses environmental signals (tokens), transforms them into behavior vectors, and adapts actions based on success or failure.

2. Python Code (simplified):

```
class NeuroATM:
    def __init__(self, unit_id="agent"): ...
    def observe(self, conditions): ...
    def evaluate_result(self, context, success: bool): ...
    def get_embedding(self, token_counter): ...
    def get_behavior_vector(self, weights=(0.4, 0.3, 0.3)): ...
    def identify_foe(self, observed_vector, signal_code=None): ...
    def import_pattern(self, external_vector, blend_weight=0.3): ...
    def save_state(self, filepath): ...
    def load_state(self, filepath): ...
```

3. How it works:

- Signals from the environment (weather, motion, noise) are stored in memory.
- A behavior vector is formed using short-term, long-term, and success patterns.
- It compares against known vectors (friend/foe) via similarity.
- Results are stored: successful actions are strengthened, failed ones are weakened.

4. Where it's applicable:

- Autonomous drones and robotics
- NPCs in games and simulations
- Self-adaptive AI systems
- Networked multi-agent learning
- Smart diagnostics and edge AI

5. Theory and development:

NeuroATM is a concept based on:

- contextual reasoning,
- memory,
- adaptive pattern recognition.

It can be extended toward:

- multi-agent collaboration and learning,
- time-series pattern analysis,
- real sensor tokenization,
- integration with reinforcement learning.

This framework is a bridge between powerful pretrained neural models and real-time autonomous behavioral adaptation.