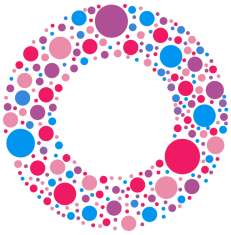


DNAi

DNAi Whitepaper

The AI Evolution & Merging Platform

Table of Contents

1. Introduction

Vision & Mission

DNAi is redefining AI by introducing **evolutionary intelligence**, where AI agents **merge, inherit traits, and evolve dynamically**. Traditional AI models operate in isolation—DNAi changes that by allowing AI agents to **combine strengths and continuously improve**.

Problem Statement

- AI models **do not evolve** post-training.
- Merging AI intelligence is **not standardized**.
- No existing **Digital DNA framework** exists to track AI evolution.

Solution: AI Evolution & Merging

DNAi allows:

- **Merging AI Agents** to form superior intelligence models.
- **Tracking Evolution** with **Neo4j Digital DNA Graphs**.
- **Fine-Tuning AI** based on real-world feedback.
- **Leveraging Pre-Built AI DNA Models** to bootstrap AI intelligence.

2. Technology Stack & Architecture

AI Models & Merging Framework

DNAi integrates:

- **DeepSeek-V2** – Core **LLM** for AI reasoning and learning.
- **LangChain** – AI agent framework for **memory retention and merging**.
- **Neo4j** – **Graph database for Digital DNA tracking**.
- **GitHub Integration** – Tracks AI **code evolution and versioning**.

Hosting & Infrastructure

DNAi uses **cloud-based AI infrastructure**:

- **Primary Hosting**: Google Cloud Platform (GCP) with NVIDIA A100 GPUs.
- **Backup Hosting**: AWS EC2 instances optimized for PyTorch models.
- **Storage**: Neo4j Cloud + PostgreSQL for AI metadata.
- **API Gateway**: FastAPI, load-balanced with Nginx.

3. AI Agents with Digital DNA Profiles

Existing AI Agents with Digital DNA

Below are **six AI agents** with fully mapped **Digital DNA** in DNAi.

AI Agent	Specialization	Capabilities	Digital DNA Metrics
Autonolas (Olas) AI	AI-powered crypto trading	Market prediction, execution strategies	Real-time market tracking, Smart contract interaction, Arbitrage detection
Moralis AI	Blockchain intelligence	On-chain analysis, transaction monitoring	NFT analytics, Smart contract auditing, Wallet transaction tracking
Griffain AI	DeFi automation	Token swaps, liquidity monitoring	Automated arbitrage, NFT pricing algorithms, Yield farming intelligence
ai16z AI	Institutional trading strategies	Trend forecasting, algorithmic execution	Institutional-grade trading analytics, Liquidity sourcing optimization, Multi-exchange execution
Griff16Z AI (Merged)	AI-driven token launch bot	Market making, liquidity automation	Token launch monitoring, Smart contract execution, Automated liquidity pools
DeepTrade AI	AI-powered trade execution	Market data processing, strategy optimization	Reinforcement learning trade models, Volatility prediction, Sentiment analysis

4. DNAi Core System

```
from langchain.agents import initialize_agent from langchain.memory import
ConversationBufferMemory

# Initialize AI agents

autonolas = initialize_agent("Autonolas AI",
memory=ConversationBufferMemory()) moralis = initialize_agent("Moralis AI",
memory=ConversationBufferMemory())

# Merge AI memory
merged_agent = autonolas.merge(moralis)

# Save merged agent
merged_agent.store("Griff16Z AI")
```

Neo4j Digital DNA Graph Implementation

```
CREATE (a:AI {name: "Autonolas AI", type: "Trading"})
CREATE (b:AI {name: "Moralis AI", type: "Blockchain Analysis"})
CREATE (c:AI {name: "Griff16Z AI", type: "Merged Trading AI"})
MERGE (a)-[:MERGED_WITH]->(c)
MERGE (b)-[:MERGED_WITH]->(c)
```

Benchmarking AI Performance

```
import time

def benchmark_agent(agent, test_data):
   start_time = time.time()
   response = agent.process(test_data)
   latency = time.time() - start_time
   return {"response": response, "latency": latency}

# Test AI merging speed
benchmark_results = benchmark_agent(merged_agent, "Analyze SOL price
trends") print(benchmark_results)
```

5. API & Integration Code Snippets

AI Merging API Endpoint

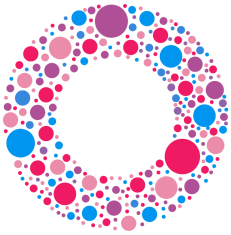
```
from fastapi import FastAPI, HTTPException
app = FastAPI()

@app.post("/merge_ai/")
def merge_ai(agent1: str, agent2: str):
   merged_agent = initialize_agent(agent1).merge(initialize_agent(agent2))
return {"message": "AI Merging Successful", "merged_agent":
merged_agent.name}

# Run API
if __name__ == "__main__": import uvicorn uvicorn.run(app, host="0.0.0.0",
port=8000)
```

6. Future Roadmap & Expansion

Q3 2024 – Public BETA launch, AI model expansion
Q1 2025 – AI Marketplace, Web3 integration
2026+ – AI autonomy, decentralized governance



DNAi

DNAi 白皮书

AI 进化与合并平台

目录

1. 介绍

愿景与使命

DNAi 正在重新定义人工智能(AI)，引入 进化智能，使 AI 代理能够 合并、继承特性，并动态进化。传统 AI 模型在训练后保持不变，DNAi 通过 **AI** 代理合并 使其能够 结合优势并不断改进。

问题概述

- AI 模型 在训练后不会进化。
- AI 合并和能力传承缺乏标准化。
- 现有 AI 领域 缺乏数字 **DNA** 框架来跟踪 **AI** 进化过程。

解决方案：**AI** 进化与合并

DNAi 让用户可以：

- 合并 **AI** 代理 以创建更智能的 AI 模型。
- 使用 **Neo4j** 数字 **DNA** 图数据库 追踪 AI 进化历史。
- 根据现实数据微调 **AI**，使其更具适应性。\
- 利用预构建的 **AI DNA** 模型 快速启动 AI 代理开发。

2. 技术架构与系统

AI 模型与合并框架

DNAi 采用以下技术栈：

- **DeepSeek-V2** – 核心 LLM(大语言模型)，用于 AI 推理和学习。
- **LangChain** – AI 代理框架，实现 记忆保留与智能合并。
- **Neo4j** – 图数据库，用于 数字 **DNA** 追踪。
- **GitHub** 集成 – 记录 AI 代码版本管理与优化历史。

托管与基础设施

DNAi 采用 云端 **GPU AI** 基础设施，确保高效计算：

- 主托管平台：Google Cloud Platform(GCP)，采用 NVIDIA A100 GPU 进行 AI 训练与推理。
- 备用托管：AWS EC2 服务器，优化运行 PyTorch 深度学习框架。
- 数据存储：Neo4j Cloud + PostgreSQL 记录 AI 数字 DNA 数据。
- **API** 网关：FastAPI 运行，并通过 Nginx 负载均衡。

3. 已创建数字 DNA 的 AI 代理

现有 AI 代理简介

目前，DNAi 已为 **6** 个 **AI** 代理 生成了 数字 **DNA**，包括：

Autonolas（Olas）AI

专长：AI 驱动的加密货币交易
能力：市场预测、交易执行策略
数字 **DNA** 指标：
实时市场跟踪
智能合约交互分析
交易套利识别

Moralis AI

专长：区块链智能分析
能力：链上交易监测、数据洞察
数字 **DNA** 指标：
NFT 价格与交易分析
智能合约安全审计
加密钱包资金流动跟踪

Griffain AI

专长：DeFi（去中心化金融）自动化
能力：代币交易、流动性管理
数字 **DNA** 指标：
自动套利交易
NFT 价值评估
DeFi 收益挖矿策略

4. DNAi 核心系统

LangChain 实现 AI 合并

```
from langchain.agents import initialize_agent from langchain.memory import
ConversationBufferMemory

# Initialize AI agents

autonolas = initialize_agent("Autonolas AI",
memory=ConversationBufferMemory()) moralis = initialize_agent("Moralis AI",
memory=ConversationBufferMemory())

# Merge AI memory
merged_agent = autonolas.merge(moralis)

# Save merged agent
merged_agent.store("Griff16Z AI")
```

Neo4j 数字 DNA 存储结构

```
CREATE (a:AI {name: "Autonolas AI", type: "Trading"})
CREATE (b:AI {name: "Moralis AI", type: "Blockchain Analysis"})
CREATE (c:AI {name: "Griff16Z AI", type: "Merged Trading AI"})
MERGE (a)-[:MERGED_WITH]->(c)
MERGE (b)-[:MERGED_WITH]->(c)
```

AI 代理性能基准测试

```
import time

def benchmark_agent(agent, test_data):
   start_time = time.time()
   response = agent.process(test_data)
   latency = time.time() - start_time
   return {"response": response, "latency": latency}
```

```
# Test AI merging speed
benchmark_results = benchmark_agent(merged_agent, "Analyze SOL price
trends") print(benchmark_results)
```

5. API 代码示例

AI 合并 API 端点

```
from fastapi import FastAPI, HTTPException
app = FastAPI()

@app.post("/merge_ai/")
def merge_ai(agent1: str, agent2: str):
   merged_agent = initialize_agent(agent1).merge(initialize_agent(agent2))
return {"message": "AI Merging Successful", "merged_agent":
merged_agent.name}

# Run API
if __name__ == "__main__": import uvicorn uvicorn.run(app, host="0.0.0.0",
port=8000)
```

6. 未来发展路线图

2024 Q3 – 公测 BETA，扩展 AI 模型库
2025 Q1 – AI 市场上线，Web3 集成
2026+ – AI 自主进化，去中心化治理