Near-Quantum Agentic AGI Training Set Based on QData and Uploaded Files

Near-Quantum Agentic AGI Training Set: Synthesis from QDataSet and DevUtility Specialized Resources

---

Introduction

The pursuit of Artificial General Intelligence (AGI) remains one of technology's grandest challenges. Increasing research points to the power of including both **agentic design principles** -where autonomous, context-aware "agent" systems drive adaptive behaviors-and *near-quantum computational paradigms*, which blend classical AI with algorithmic insights and data structures emerging from quantum information science[1]. Producing an effective **training set for near-quantum agentic AGI** is a nontrivial task, as it requires synthesizing expertise from quantum machine learning, robust agent architectures, reinforcement learning, and structured data preparation.

This report presents a comprehensive **training set framework** in Markdown, designed by integrating the architecture and content of the GitHub eperrier QDataSet[3][4], ARC-AGI-2 and associated guides[6][7], and a corpus of DevUtility specialized training files, including domain-specific terms, code examples, utility functions, and agentic principles from CodeReaver & CodeRebel, NNMM-AA neural agent architecture, and concept dictionaries.

Below, a detailed, paragraph-driven synthesis interleaves key concepts, web-sourced insights on quantum/agentic AGI, and explicit source mappings to uploaded documents. The end result is a logically organized, extensible Markdown training set, equipped to advance AGI research.

Summary Table: Key Concepts and Source Mappings

Concept/Module	Source(s)	Relevance
QDataSet Architecture	QDataSet GitHub, README, arXiv paper[3][4]	Quantum datasets for ML, simulation of qubit systems.
Near-Quantum Computation Paradigms	UChicago, Gino Volpi, Nature, Springer[9][1]	Quantum features, superposition, industry trends.
Agentic AGI Principles	Retell AI, Future AGI, Analytics Vidhya[11]	Agent frameworks, autonomy, adaptability, RL.
ARC-AGI-2 Benchmark	ARC-AGI-2 GitHub, Prize Guide, Site[6][7]	Testing general fluid intelligence in AI.

NNMM-AA Explained	nnmm-aa-explained.txt, NNMM-AA repo[12]	Advanced custom NN training loop, agentic coding.
DeadSnakes PPA	deadsnakesppa.txt, AskUbuntu, Launchpad[14]	Python environments for utility and AGI experimentation.
DevUtility Training Set Structure	DevUtility files, AndroidV2.5 terms, DevUtility GitHub[16]	Modular, agent-centric development utilities.
K/C/Near-Quantum Specialized Modules	K-Near-Quantum, C-Near-Quantum, CodeReaver, CodeRebel, specialized.txt	Agentic code, utility specialization, customization.
Online Example Modules	FROM-ONLINE-EXAMPLE.txt	Exemplar utility module creation, best practices.
Markdown Dataset Structuring	Embyr, DocScraper, HuggingFace, Markdown Tutorials[18][19]	Dataset cleaning, RAG optimization, Q&A prompt design.
RAG for Quantum/AGI Training	Arxiv 2508.21097, GitHub ICP, DocScraper[17]	Retrieval-Augmented Generation, code search, model utility.
Multi-Agent RL Integration	Prime Intellect, Future AGI, Xaxis/agentic-agi[22][23]	Multi-agent orchestration, open RL environments.
Android, Utility Dictionaries	AndroidV2.5_Terms, Android Developer[24]	Task terms, UI, and contextual data structuring.

As seen above, this training set synthesis draws from a spectrum of resources-quantum data, agent frameworks, specialized AGI code, RL benchmarks, and formatting utilities-to build a unified curriculum for near-quantum agentic AGI.

Section 1: Quantum Datasets and Near-Quantum Computational Paradigms

1.1 QDataSet: Quantum Dataset Architecture and Role in AGI Training

The QDataSet, introduced by Perrier, Youssry & Ferrie[3][4], forms a foundational component for quantum machine learning (QML) research and directly informs the agentic AGI dataset's quantum content. QDataSet comprises 52 simulation-based datasets of **one- and two-qubit systems**, tracking their evolution under various noise profiles and classical/quantum controls. Importantly, these datasets focus on problems directly relevant to both industrial and theoretical quantum computation, such as **quantum control, tomography, and noise spectroscopy**.

The structure and naming conventions adopted in QDataSet ensure *interoperability* and *portability* across standard machine learning and quantum computing platforms, with support for direct integration into Python ML workflows. By encapsulating noise models, entanglement, and quantum dynamics in labeled data, QDataSet enables machine learning practitioners to

develop and test algorithms not just in classical domains, but also for quantum-enhanced reasoning-of critical importance for training agent systems to deal with *uncertainty and non-classical information spaces*.
Meanwhile, QDataSet's companion workbooks and examples offer a roadmap for not only *data loading and preprocessing* but also advanced *optimization strategies*-all highly instructive for AGI pipelines seeking parallel quantum-inspired representations. The direct use of these datasets in the training set supports realistic quantum/near-quantum behaviors in agentic AI.

1.2 Near-Quantum Paradigms in AI: Industrial and Research Trends

Quantum computation offers highly distinct paradigms from classical computation: *superposition*, in which quantum bits (qubits) can be in simultaneous logical states, and *entanglement*, which allows distributed correlations that classical systems cannot emulate[8][1].
While truly scalable quantum computers remain on the horizon, the current "near-term" era is dominated by **noisy intermediate-scale quantum (NISQ)** systems: stable enough for meaningful research, yet prone to error, requiring robust error-mitigation and hybrid computational strategies.
Industry investments have soared, with IBM, Google, and sector startups rapidly developing quantum-centric supercomputers and cloud APIs, now widely accessible to researchers.
Applications in **pharmaceuticals, finance, cybersecurity, and scientific modeling** have highlighted the transformative potential of quantum computing. Notably, government and industry expect full-scale adoption as early as 2035, underscoring the urgency to prepare compatible agentic training resources now[8].
Recent advances in **quantum machine learning** (QML) show the field is moving beyond pure theory. Practical QML leverages quantum-inspired representations (e.g., quantum vector support, variational circuits, quantum neural networks) to accelerate learning, enhance optimization, and facilitate tasks intractable for classical ML. The QDataSet's focus aligns with these directions, generating training challenges directly applicable to emerging QML models[25][26].

1.3 Hybrid and Model-Driven Code Generation for Quantum/Agentic AI

A critically enabling technology is **retrieval-augmented generation (RAG)**, in which LLMs are supplemented with code, documentation, and domain examples during code synthesis, especially important for **quantum software engineering** and hybrid agent systems[17]. RAG pipelines for QML code have been shown to yield dramatic improvements in correctness and maintainability of quantum utility functions and APIs. Integrating this approach in our training set design allows for *dynamic agent utility module embedding*, where agents can access, retrieve, and execute contextually relevant quantum routines during RL loops or data processing.

Section 2: Agentic AGI: Principles, Architectures, and Agent Frameworks

2.1 Defining Agentic AI and Its Distinction from Narrow AI

Agentic AI and AGI are related but not synonymous. While **narrow AI** focuses on specific, pre-defined tasks, **agentic AI** is characterized by *autonomy, adaptability, goal orientation, and context awareness*-qualities essential for AGI that aspires to human-like generalization[11]. Agentic frameworks empower software entities, "agents," to make independent decisions, learn from their environments, and pursue complex goals that may involve built-in or dynamic strategies. Key features of agentic AGI relevant for training set design include:

- **Autonomy**: Operation with minimal external intervention; agents self-direct actions toward assigned tasks.
- **Goal orientation**: Agents dynamically prioritize and adjust strategies according to current objectives.
- **Adaptability**: The capacity for reflection and behavioral revision, underpinning continual learning.
- **Context awareness**: Perception and understanding of dynamic, multi-modal environments, essential for RL and general decision-making.
- **Collaborative and multi-agent orchestration**: Hierarchical or peer-to-peer coordination among agents to break down or sequence subtasks[22].

The distinction is critical for AGI datasets-while data must train basic task skills, it must simultaneously provide *meta-level reasoning, self-improvement, and adaptive context handling*.

2.2 Modern Agentic Frameworks and RL in AGI

Recent years have seen the emergence of production-ready **agentic AI frameworks**, such as LangChain, CrewAI, and Microsoft's AI kernel, which accelerate the development of agentic workflows[11][27]. Best practices in these frameworks include:

- *Modular design*: Separating agent "brains" (decision logic) and "tools" (APIs/utilities) for maximal flexibility.
- *Task and environment interfaces*: Allowing agents to interact with, manipulate, and learn from RL environments.
- *Reflection and self-improvement*: Architectures that encourage agents to evaluate outcomes, learn from mistakes, and refine behaviors.

The **Relevance-Aware Agentic (RAA) system**[23], for instance, orchestrates specialized agents within a multi-agent RAG (retrieval-augmented generation) framework, conflating emergent collective reasoning, self-correction, and context-driven adaptation-a computational analog to fluid human intelligence.

Recent platforms such as the RL Environments Hub[22] offer extensive open RL environments for AGI training, allowing agents to learn through experimentation and reinforce policies via reward

signals. These environments are essential for our dataset: they foster robust, adaptable AGI performance.

Section 3: DevUtility, Specialized Utility Modules, and Concept Dictionaries

3.1 DevUtility and Agent Utility Datasets

The **DevUtility** family of datasets and utilities, as represented in the training corpus (see DevUtility Trainingset_250807_151722.txt, DevUtilityAndroidV2.5 Terms_Concepts_Dictionary, and associated specialized modules)[16], supports the construction, adaptation, and testing of AGI agent code and utility functions. In the context of near-quantum AGI, DevUtility training sets encompass:

- **Custom code interfaces**: For Android and multi-platform device interaction, including split-screen utility, code editing, and context manipulation.
- **Dictionary modules**: Defining specialized terms (activity, CTA, density-independent pixels, intent) to ensure semantic clarity and standardized data interchange.
- **Modular plug-and-play agent functions**: Allowing for rapid utility deployment in agent or RL platforms.

These datasets integrate both agents' internal knowledge representations and the utility modules required for interaction with diverse operating environments-a critical AGI need.

3.2 K/Near-Quantum and CodeReaver & CodeRebel Specialized Modules

The uploaded set includes **K-Near-Quantum** and **C-Near-Quantum** DevUtility modules, as well as CodeReaver and CodeRebel content-these represent **domain-specific agent utility extensions** that focus on optimization, code efficiency, and platform adaptivity. Key features from these modules (as inferred from file content and available documentation) include:

- **Code optimization and transformation tools**: For both classical and quantum utility functions.
- **Platform abstraction**: Routines that adapt agent behavior/function across operating systems (Android, Linux, Windows).
- **Advanced debugging and reflection hooks**: For agent introspection and real-time utility self-update.
- **Quantum logic wrappers**: Functions for noise mitigation, parameter tuning, and quasi-quantum effect simulation (mirroring techniques documented in recent quantum robustness literature[28]).

This modular approach facilitates retrieval-based and context-driven agent operation, minimizing the codebase size while maximizing relevance realization.

3.3 NNMM-AA Explained: Advanced Neural Agent Coding

The content sourced from **nnmm-aa-explained.txt** and the public **NNMM-AA** repo[12]showcase sophisticated custom neural architectures integral to an agentic AGI training set. Highlighted techniques include:

- **Custom residual blocks and dense layers**: Enhancing signal propagation, robustness, and deep structure learning-critical for both classical and quantum model transfer.
- **Distributed learning strategies**: Enabling agents to scale learning process across environments, echoing the needs of RL and multi-agent cooperation.
- **Custom training loops with adaptive learning rate schedulers**: Improving convergence and generalization.

Detailed logging (e.g., via TensorBoard) and architecture/weight export allow for streamlined evaluation and transfer between agent roles or simulation environments.

Section 4: Practical Data Integration, Python Environments, and DeadSnakes PPA

As agentic AGI platforms often require dynamic code execution and modular utility loading at runtime, stability and extensibility of the developer environment are critical. Python's centrality to ML and agentic platforms demands practical, properly managed system environments. **DeadSnakes PPA** emerges as a solution to obtain up-to-date Python interpreters-critical for AGI agents running cross-version or emergent code[14]. Notably:

- The DeadSnakes repository provides a spectrum of supported Python releases for Ubuntu systems, which is essential for dependency isolation and testing across Python versions.
- The PPA is maintained by trusted developers and is widely used by researchers, though caution is still urged for mission-critical or highly secure deployments.
- The PPA includes core and development libraries, enabling AGI agents to take advantage of evolving Python features and syntaxes.

Within the training set, including modules for *OS version detection, dynamic Python environment selection, and secure package installation* aligns with best practices noted in expert-answered Q&A threads and platform maintainers' guides.

Section 5: ARC-AGI-2 Benchmark Tasks and Relevance to General Intelligence

The **ARC-AGI-2** benchmark is the gold standard for testing general AI, designed to probe *fluid intelligence, adaptive skill transfer, and compositional reasoning*[6][7].
Core features for integration in the training set:

- **1,000 public training tasks and 120 evaluation tasks**: A mixture of simple and highly compositional reasoning problems, formatted as grid color manipulations and rule discovery.
- **Skill acquisition, not just skill demonstration**: Each ARC-AGI-2 task is solved by applying minimal but expressive core knowledge priors.
- **JSON-based structure**: Facilitating easy import as RL environment "worlds" for agents.
- **Multi-tier evaluation**: Supporting both open benchmarking and secure, leak-proof competition environments.

Incorporating *snippet examples and API wrappers* for ARC-AGI-2 task ingestion and agent interaction, as part of the training set, ensures AGI agents are benchmarked and optimized to industry and research-leading standards.

Section 6: Markdown Training Set Structuring and Data Preparation

Modern training systems require data in *highly structured, clean, and metadata-rich formats*. Markdown, with its balance of readability and machine processability, is a preferred foundation [18][29].

Key best practices for the near-quantum agentic AGI training set:

- **Sectioning and headings**: Each concept, utility module, agent role, quantum function, and task pattern has its own top-level Markdown header.
- **Definitions and code blocks**: All core terms, dictionary entries, and exemplar modules are distinctly defined, sometimes with code snippets when appropriate.
- **Example prompt-completion pairs**: For RAG and RL, clear context-completion examples demonstrate intended behaviors.
- **Structured tables**: Used for concept mappings, code utility indexes, and agent coupling relationships.

Cleaning operations-including removal of extraneous tags, unification of casing, and elimination of redundant metadata-are employed, while context-preserving segmentation prepares data for both human and LLM consumption.

Section 7: Retrieval-Augmented Generation and Multi-Agent Orchestration

Retrieval-Augmented Generation (RAG) further enables true agentic performance in AGI by giving agents on-demand access to documentation, code, and working examples relevant to their current context[17].

Key RAG features embedded in the training set:

- **Sample-based prompting**: Embedded code, protocol, and definition prompts for live agent code synthesis and task performance.

- **Multi-agent context sharing**: Multi-agent workflows, as described in Future AGI and "Relevance-Aware Agentic" frameworks[23], are represented through markdown-based interaction protocols for role assignment, subgoal decomposition, and hierarchical RL.
- **Reflection and self-correction hooks**: Agent modules are equipped with reflection and strategy update mechanisms by design, echoing contemporary agent factory design patterns [27].

These mechanisms are indispensable for benchmarking and real-world AGI deployment.

Structured Training Set in Markdown

What follows is an extensible, modular **AGI Training Set in Markdown format**. Each module or concept includes context, intended use, and origin for clarity.

NEAR-QUANTUM AGENTIC AGI TRAINING SET

Quantum Datasets and Computational Paradigms

QDataSet: Quantum Benchmark Datasets

Summary:
QDataSet provides 52 high-quality, simulated datasets for quantum machine learning. Each dataset tracks one- or two-qubit system evolution under various noise and control settings, supplying inputs ideal for quantum-enhanced agent training.
Fields:

- Qubit count (1 or 2)
- Initial states (Bloch vector)
- Quantum control parameters (pulse, field, etc.)
- Noise profile (none, dephasing, depolarizing, amplitude damping)
- Observed outputs (state vectors, measurement outcomes)

Reference: QDataSet GitHub, arXiv[3][4]

Near-Quantum Computation

Definition:
A computational paradigm blending classical AI with algorithmic patterns, data structures, and mathematical constructs emerging from quantum principles. Central elements include:

- *Superposition*: Parallel computation over multiple logical states.

- *Entanglement*: Passing correlated information between agents or functions-used in multi-agent orchestration.
- *Noise modeling and resilience*: Agents learn to operate under uncertainty and stochasticity, mapping quantum error behaviors onto classical structures for improved adaptability.

Industrial Context:
IBM, Google, and academic partners are leading the way in industrial QML and near-term quantum computing research. Governmental strategies expect workforce and mainstream deployment by 2035[9].

Agentic AGI Principles and Frameworks

Agentic AI and General Intelligence

Definition:
Agentic AI systems feature autonomous, adaptable agents that reflect, plan, and act on goals within complex, potentially unknown environments.
Core Principles:

- **Autonomy**: Agents act independently to pursue goals.
- **Goal-Orientation**: Each agent is assigned, or develops, objectives.
- **Adaptability**: Behavior changes as agents learn from experience (continuous RL).
- **Context-Awareness**: Sensitivity to environment and state.
- **Collaborative Orchestration**: Multi-agent structures enable complex task decomposition and role assignment.

References: Retell AI, Future AGI, Analytics Vidhya[11][23][22]

Modern Agentic Frameworks

Exemplar Architectures:

- *LangChain*: Modular agent “brain” and “tool” composition.
- *Microsoft Semantic Kernel*: Reflection and planning.
- *Relevance-Aware Agentic (RAA)*: Multi-agent orchestration with live context recognition and RAG-pipelined retrieval.

Best Practices:

- Reflexivity: Agents regularly reflect on their strategies and outcomes.
- Plug-and-play environments: Agents adapt to new RL or task environments with minimal overhead.

ARC-AGI-2: Generalization Benchmark

Overview

- **Task Format**: Visual/compositional reasoning in grid format (JSON data), requiring the agent to abstract and transfer skills.
- **Dataset**: 1,000 public training tasks, 120 evaluation tasks.
- **Benchmarking**: AGI systems must achieve at least 85% accuracy on private sets to rival human performance.

Formatting Example:

```
{
"train": [
{"input": [[1,0,0],[0,0,1]], "output": [[1,1,1],[1,0,1]]}
],
"test": [
{"input": [[...]]}
]
}
```

References: ARC-AGI-2 GitHub, Guide[6][7]

Agent Utility Modules and Dictionaries

DevUtility: Specialized Utility Functions

Core Functions:

- *Activity management*: For Android or desktop applications.
- *Split-screen support*: For multitasking or multi-agent environments.
- *Manual code editing*: Agents can interactively modify deployed code.
- *Platform abstraction*: Detect OS, set relevant paths, and invoke platform-specific handlers.

Reference Docs: DevUtilityAndroidV2.5, DevUtility Specialized, Android Developers[16][24]

Concept Dictionary

Term	Definition
Activity	An application UI and action set (Android, UI)
CTA	Call to Action; user's goal or prompted action
Intent	Signal for inter-app communication
Density-Independent	Pixel unit for consistent UI across different device screens
Task	Sequence of activities to accomplish a goal (stack-based)

Reflection Module	Agent function for self-assessment
Quantum Control	Procedures that manipulate the quantum state

Extend this dictionary for any relevant platform-, agent-, or RL-specific terms.

Specialized Near-Quantum, K/C-Near-Quantum, CodeReaver, CodeRebel Modules

Quantum Agent Utility

- **Noise Modeling**: Functions for generating, simulating, or compensating quantum noise in agent state-tracking.
- **Quantum Gate Abstraction**: Agent-accessible APIs wrap gate operations for simulation or quantum-emulated RL.

Code Optimization Utilities

- **Platform Detection**: Dynamic selection and instantiation of platform, language, or interpreter.
- **Transformation**: Runtime conversion between classical and quantum code modules.

Adaptive RL Agent Wrappers

- **Reflection and Self-Correction**: Integrated feedback routines.
- **Dynamic API Loading**: Retrieval-based importation of best-suited routines based on current subtask.

Neural Network Mind Map (NNMM-AA): Agentic Model Code

Custom Dense Layer

```
class CustomDense(tf.keras.layers.Layer):
def __init__(self, units, activation=None):
...
def call(self, inputs):
...
```

Residual Block

```
class ResidualBlock(tf.keras.layers.Layer):
...
```

def call(self, inputs):
...

Custom Training Loop

def custom_train_loop(model, dataset, epochs):
...

These elements allow agents to instantiate and adapt NNs suited to their current environment, integrating downstream quantum modules or classical RL routines.

Python Environment Management: DeadSnakes PPA Integration

Guidance

Install Specific Python Version:

- Detect optimal Python version for context (Ubuntu 20.04, 22.04, etc.)
- Use DeadSnakes PPA as fallback for unsupported native versions

sudo apt install software-properties-common
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt install python3.12

Caution: Prefer use in isolated environments; be mindful of stability for production.
References: AskUbuntu, Launchpad[14]

Online Example Modules

Exemplar Utility: Contextualized Completion

- Modular code for environment detection (OS, shell)
- Example API request handler (for RL agent tool use)
- Dynamic module loader template

Markdown Structuring and Data Preparation

Data Cleaning Procedures

- Remove extraneous tags or formatting
- Unify casing across all tokens
- Tokenize and segment for Q&A-based prompt-completion sets
- Retain contextual continuity for multi-step tasks

Reference Docs: Embyr, DocScraper, HuggingFace AGI Protocols[18]

Retrieval-Augmented Generation (RAG) and Multi-Agent Coordination

RAG Integration

- Index code samples, documentation, and quantum utility functions
- Use embedding-based retrieval via vector DB for contextually relevant code/data

Multi-Agent Orchestration

- Enable “planner-executor” role assignment
- Define “task”, “subtask”, and “reflection” modules

Example Markdown:

```
## AGENT: QUANTUM-PLANNER

- GOAL: Optimize quantum control routine for task X
- TOOLS: retrieve quantum_control_fn, simulate_noise_profile
- REFLECTION: Evaluate improvement over baseline; update plan if suboptimal

## AGENT: EXECUTOR

- Receives plan, executes code, returns result
```

References: Future AGI, Prime Intellect, Xaxis agentic-agi[23][22]

Sample Prompt-Completion Pairs for Q&A/Educational Protocols

Prompt	Completion
“Given noisy amplitude damping in a qubit, recommend the best control strategy for RL agent optimization.”	“Apply amplitude damping compensation by adjusting Rabi pulse timings, referencing q_control_amplitude.py. Feedback to RL loop.”
“How can AGI agents decompose a multi-agent orchestration task involving symbolic ARC manipulation?”	“The planner agent assigns subgrids; executors handle localized pattern identification. Feedback mediates final grid assembly.”

Reference: HuggingFace AGI protocols, Educational Q&A datasets[18].

Data Integration Best Practices

Key Guidelines

- *Eliminate data silos*: Link agent utility datasets, quantum data, and RL tasks into a single workflow.
- *Enable table-based and context-driven retrieval*: Index datasets with structured metadata.
- *Prioritize modularity*: Each Markdown file/section should operate independently and support dynamic agent retrieval and embedding.

Reference: Estuary Blog on Data Integration[30].

Data Structuring Table

Section/Module	Markdown Header	File Origin	Agentic/Quantum Role
Quantum Dataset	## Quantum Datasets	QDataSet, arXiv, README	QML, RL, uncertainty
Agentic Utility	## DevUtility	DevUtilityAndroid, Specialized files	Task, platform interface
Neural Architecture	## NNMM-AA	NNMM-AA, explained.txt	Model, loop, adaptation
Dict/Terms	## Concept Dictionary	AndroidV2.5_Terms, DevUtilityDict	Semantic clarity
Multi-Agent Protocol	## Agent Protocols	K/C-Near-Quantum, CodeReaver/Rebel	Orchestration, retrieval
Benchmark Tasks	## ARC-AGI-2	ARC-AGI-2, Prize Guide	Evaluation, skill metrics
RL Environments	## RL Integration	RL Hub, Future AGI, Prime Intellect	Training environments
Data Structure	## Formatting Guide	Embyr, DocScraper, HuggingFace	Clean, contextual Syntax

---

Conclusion

A near-quantum agentic AGI training set demands the conscientious curation and integration of quantum simulation data, modular agent utility code, multi-agent orchestration patterns, and robust RL environments. This report demonstrates how to synthesize these diverse resources into a single extensible Markdown set: each module is intentionally designed for plug-and-play usage by both human practitioners and AGI systems, with built-in metadata, role/context cues, and quantum/agentic transformation hooks.

By merging **QDataSet's quantum data**, **ARC-AGI-2's general intelligence tests**, specialized **DevUtility modules**, advanced **agentic neural architectures**, RAG-based sample integration, and robust Python environment support, this blueprint lays the foundation for a flexible, scalable, and future-proof AGI curriculum.
Users and AGI systems are encouraged to expand, customize, and retrieve modules as RL environments and quantum platforms advance-ensuring the dataset's continued relevance and capability expansion as AGI and quantum technologies mature.

References (30)

1. *Quantum Machine Learning for AGI: Redefining Intelligence Through ...*. https://link.springer.com/content/pdf/10.1007/978-3-031-87931-9_13.pdf?pdf=inline%20link
2. *Environments Hub: A Community Hub To Scale RL To Open AGI*. https://www.primeintellect.ai/blog/environments
3. *GitHub - Xaxis/agentic-agi*. https://github.com/Xaxis/agentic-agi
4. *Android Terminology - Coders Helpline*. https://www.codershelpline.com/courses/proglanguages/android/android-terminology/
5. *Training deep quantum neural networks - Nature*. https://www.nature.com/articles/s41467-020-14454-2.pdf
6. *Toward Trainability of Deep Quantum Neural Networks*. https://arxiv.org/pdf/2112.15002
7. *Agent Factory: The new era of agentic AI-common use cases and design ...*. https://azure.microsoft.com/en-us/blog/agent-factory-the-new-era-of-agentic-ai-common-use-cases-and-design-patterns/
8. *Provably Robust Training of Quantum Circuit Classifiers Against ...*. https://arxiv.org/pdf/2505.18478
9. *Glossary* . https://developer.android.google.cn/design/ui/mobile/guides/foundations/glossary?hl=en
10. *Practical AGI Through Educational Protocols: Teaching Structural ...*. https://huggingface.co/blog/kanaria007/practical-agi-through-educational-protocols
11. *Markdown Tables Syntax - Tutorial*. https://htmlmarkdown.com/syntax/markdown-tables/
12. *DocScraper - GitHub*. https://github.com/eagurin/docscraper
13. *python - Why cannot add PPA deadsnakes? - Stack Overflow*. https://stackoverflow.com/questions/66597894/why-cannot-add-ppa-deadsnakes
14. *Top 7 Frameworks for Building AI Agents in 2025 - Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2024/07/ai-agent-frameworks/
15. *NNMM-AA/README.md at main · spiralgang/NNMM-AA · GitHub*. https://github.com/spiralgang/NNMM-AA/blob/main/README.md
16. *ARC Prize - Guide*. https://arcprize.org/guide
17. *ARC-AGI-2*. https://arcprize.org/arc-agi/2/

18. *Challenges and Opportunities of Near-Term Quantum Computing Systems*. https://arxiv.org/pdf/1910.02894.pdf
19. *Quantum Paradigm Shift* . https://professional.uchicago.edu/stories/quantum-science-networking-and-communications/quantum-paradigm-shift?language_content_entity=en
20. *QDataset: Quantum Datasets for Machine Learning - arXiv.org*. https://arxiv.org/pdf/2108.06661v1
21. *[2108.06661] QDataset: Quantum Datasets for Machine Learning*. https://arxiv.org/abs/2108.06661
22. ⬇ *The Only Markdown Cheatsheet You Need - GitHub*. https://github.com/im-luka/markdown-cheatsheet
23. *Data Integration: Complete Guide to Architecture, Tools, Methods, and ...*. https://estuary.dev/blog/data-integration/