# How we go from AI to AGI (ai2agi)

Ai70000, Ltd.

Alex Osterneck, CLA, MSCS

Date: March 2nd, 2025

## Abstract

*Our most detailed information of Babbage's Analytical Engine comes from a memoir by Lady Lovelace (1842). In it she states, "The Analytical Engine has no pretensions to originate anything. It can do whatever we know how to order it to perform" (her italics). This statement is quoted by Hartree (1949) who adds: "This does not imply that it may not be possible to construct electronic equipment which will 'think for itself,' or in which, in biological terms, one could set up a conditioned reflex, which would serve as a basis for 'learning.'* [70].

That's Alan Turing in 1950, quoting Hartree in 1949, quoting Lovelace in 1842, in reference to Charles Babbage's Analytic Engine (the first Turing Complete general-purpose computer,) first described in 1837 [70].

This modern historical context of historically important educated visionaries who formulated compute models towards future Artificial Intelligence, and, Artificial General Intelligence is still relevant.

As with Tim Berners-Lee in 1989. and then CERN in 1993 placing World Wide Web software into the public domain, there are 'pre' and 'post' dates for the dawn of the internet. In the public vernacular, pre-internet was prior to April 1993, and, post-internet after April 1993 [77].

Fast forward to OpenAI's release of ChatGPT in November 2022.

In the public domain, pre-Ai was prior to November 2022, and post-Ai after November 2022.

Since the November 2022 explosion of ai_LLMs into the mainstream, this once in a life-time phenomenon has already made a multi-trillion-dollar impact on the U.S economy.

With this increased proliferation of ai_LLM's, led by openAI and other LLM's, a tectonic shift is under way with extraordinary implications for human progress, ai_LLM's are one of the greatest inventions in world history.

But, AI as we know it in 2025 is not true AGI.

Prior to this recent explosive wave, AI went through stages of innovation, and advanced, in part, the result of Alan Turing (1939), McCulloch & Pitts (1943), Claude Shannon (1948), John McCarthy (1955), Arthur Samuel (1956), Newell & Simon (1957), Marvin Minksy (1960), Terry Winograd (1970), Ben Goertzel (2002) (who, with P. Voss, and S. Legg re-coined the term 'Artificial General Intelligence') [1], and Russel & Norvig (2015). Many other AI / AGI pioneers have made profound impacts on the theory, implementation, and establishment of AI [8][28][45][70][71][72][109][111].

This paper differentiates AI from AGI, in definition, and chronology, with AGI commencing in 2002 once the original vision of AI was re-articulated as 'Artificial General Intelligence'[1].

Just as the pioneers from 1939 through 2015 advanced AI, so too the pioneers since 2002 have advanced AGI, with obvious overlap.

S. Altman (OpenAI), D. Hassabis (Google DeepMind), I. Sutskever (OpenAI / SuperIntelligence), D. Amodei (Anthropic), E. Musk (xAI), Y. LeCun (Meta), G. Hinton (Google), Y. Bengio (Mila), B. Goertzel (SingularityNET), and P. Voss (AdaptiveAI), amongst other visionaries, academics, and entrepreneurs, have made substantial contributions toward AGI (in overlap with AI.) These lists aren't exhaustive and it's important to recognize the limitations of any such lists because there are many others who've made major contributions to AI and AGI other than what's mentioned here. But regardless of definition or chronology, the advances of AI in the public-domain since November 2022, primarily from ai_LLM's, has been spectacular, and the future is now [5][12][24][32][35][38][43][75].

At the same time, the further we look to the future, it must also be considered true AGI may never become reality. And for that reason the extraordinary ability, power and abilities of ai_LLMs should be recognized as a once in a lifetime advancement for humanity, for which their capabilities cannot be overstated.

If or when AGI is created, it will most likely be in form and substance other than the proposed blueprint in this research-project paper, however, perhaps some of the ideas presented might contribute in some small way to the eventual realization of true AGI. There are many other well-funded, established projects advancing true AGI, which have multi-decade track-records, massive resources, and highly-skilled team-members dedicated to the goal. This research-project paper should be considered a possible, incremental potential blueprint of one of many approaches, and not a substitute for other existing projects. That said, if some, (or all,) the schematics might be implemented in the creation of true AGI in any form, it will have assisted in the goal of creating true AGI.

***AI vs. AGI***

| Feature | ***AI*** (Pre-2002 and present) | ***AGI*** (Post-2002 and present) | |
|---|---|---|---|
| Core Concept | ***Specialized intelligence for specific tasks*** | ***Generalized intelligence, human-cognitive abilities*** | |
| Historical Context | ***Early development, focus on algorithms and rule-based systems*** | ***Re-articulation of original AI vision, emphasis on broader cognitive functions*** | |
| Primary Goal | ***Automation of specific tasks*** | ***Comprehensive cognitive abilities*** | |
| Current State (2025) | ***Advanced AI, including ai_LLMs, excelling in specific domains*** | ***Aspiring to achieve human-level cognitive capabilities*** | |
| Focus | ***Narrow AI, task-specific*** | ***Broad AI, cognitive abilities*** | |

**TABLE OF CONTENTS**

## Introduction

With the global push to implement AI, pathways for creating a blueprint for AGI have emerged.

One possible pathway which might be one step of many towards a blueprint for AGI is:

**AGI ≈ f ( G_enhanced, R_enhanced, ai_LLM, DTB, CV_Adapt )**

Each component is described, with notation, and pseudo-code, below.

This is a continuation of research started in 2023, towards a possible incremental advancement towards the foundation for true AGI within ~7 years time, or, by 2032.

While the G_enhanced, R_enhanced, and ai_LLM components are established within the public domain and have been extensively studied, the Digital Twin Brain (DTB) and Computer Vision Adaptation (CV_Adapt) (who's main function is as the central-processor, not as the typical role of actual vision,) components represent relatively newer areas for research. CV_Adapt, functioning as the system administrator to process tensor outputs from the other components, presents a unique approach.

And this, and other high-compute approaches are why the development and deployment of new AI data center infrastructure in 2025 and 2026 underscores the urgency of advancing these research areas. The substantial physical infrastructure and energy requirements for these projects necessitates immediate and large-scale development, on the potential magnitude of $500-Billion in the United States alone. (Strubell et al., 2019) [110], (OpenAI, 2025) [111].

The compute approaches and their application towards ai2agi presented will be mostly familiar to individuals with a master's or doctoral degree in computer science or AI engineering. For these readers, a focus on the DTB, CV_Adapt, and pseudo-code may be most efficient. While the DTB component draws heavily from neuroscience, its presentation is from the perspective of a master's-level computer science research project, and therefore in some of the equations in that section placeholders were used with the understanding those equations would require further computational-notation and integration at the level of neuroscience phD. (Dayan & Abbott, 2001) [49] , Kandel, et al. (2013) [83].

So upon further development and integration of the DTB, advanced computational notation would be a logical next step. This research project paper adopts a computer science and AI engineering perspective, which primarily informs the approach to the DTB component, and builds on AI engineering research from 2023, conducted just prior to the widespread adoption of current AI large language models (ai_LLMs), and initially focused on natural language processing (NLP) (Vaswani et al., 2017) [101], (Zhu et al. (2023) [112], Xiong et al (2023) [113] Fekonja et al. (2024) [114], Beniguev et al. (2021) [115].

The literature review supports the following thesis: (a) ai_LLMs will not be the sole or primary component of true artificial general intelligence (AGI), and (b) the transition from AI to AGI will involve multiple components, including ai_LLMs. (Marcus, 2020) [102],

*Pseudo*-code is included for components_1, _2, _3, _4 and _5. *Source*-code is available at GitHub.

## Overview of AGI components (see Appendix 2)

The four components: **e_Generalization, e_Reasoning, ai_LLM,** and **DTB** are specific, dynamic inputs, as tensors, to **CV_Adapt**, individually or in groups.

**1. Data Integration:**

**Dynamic Input**:

- **e_Generalization** and **e_Reasoning**: components provide high-level, abstracted tensors representing learned patterns and logical structures, fed to **CV_Adapt** as input and for generalized reasoning / deep logical inference.
- **ai_LLM**: outputs used intermittently, providing context / large-scale linguistic / cognitive insights, fed into **CV_Adapt** when language understanding or knowledge retrieval required.
- **DTB (Digital Twin)**: real-time, detailed simulation model of system/environment used for dynamic state representations.

**Flexible Input**:

- **Group 1**: **e_Generalization** and **e_Reasoning** feed together when high-level reasoning / pattern recognition needed for complex tasks.
- **Group 2**: **ai_LLM** processed independently when language-specific, textual reasoning is critical, feeding **CV_Adapt** with contextual language inputs.
- **Group 3**: **DTB** real-time, environmental data in tandem with other components for decision-making real-world adaptation / simulation.

**2. Processing:**

- **Parallel Computing**: Distributes processing tensors (individually / combination) across GPUs/TPUs.
- **CV_Adapt**: central processor, as GPU, accepts combinations of tensors from other components based context and interprets and adapts to input.

**3. Model Design:**

- **e_Generalization** and **e_Reasoning** combination for high-level abstraction and logic processing to guide **CV_Adapt** to make decisions on learned patterns / reasoning.
- **ai_LLM** specific moments when linguistic or cognitive knowledge required.
- **DTB** environmental interaction / real-time simulation for **CV_Adapt** to process / adapt dynamically.
- **Dynamic Learning**: **CV_Adapt** continually refines itself from input data streams.

**4. AGI Output:**

- **Raw Data Output**: After processing, **CV_Adapt** generates tensor data as output, which serves as the "DNA" for AGI processes, output generated from combination of reasoning, language understanding, generalization, real-time simulation data.

**G_enhanced / (e_Generalization)**

*"It was obvious in 1971 and even in 1958 that AI programs suffered from a lack of generality. It is still obvious, and now there are many more details." John McCarthhy (1971).* [78].

META-LEARNING

Sukhobokov, et. al, (2024) [2], differentiates the main functioning of intelligent-information-systems as those of cognitive-architectures (representation of knowledge,) and their opposite, known as expert-systems. Of the 140 cognitive-architectures reviewed, a final list of 42 cognitive-architectures were filtered as potentially enabling AGI. Functional components of the architectures included, in part, consciousness, subconsciousness, emotional management, ethics, social interaction, reflection, self-development, and meta-learning, for which the authors propose their own generalized architecture as a prototype as the basis to develop AGI prototypes [2]. Ultimately the authors identified only 4 architectures which featured a 'consciousness' (control of the agent's current actions,) (requiring at a minimum: self knowledge-monitoring-awareness-informing,) with a minimum of: a self-component for building AGI: LIDA, ICOM & MBCA, Clarion, and ISAAC. The authors similarly extracted architectures which met minimum requirements for: subconscious, worldview, and reflection, as functional-blocks, inherent in humans [2].

In Cichocki and Kuleshov (2020) [3], social, emotional, attentional and ethical intelligence for AGI were also analyzed and presented with working definitions via 'multiple intelligence,' and multi-agent systems having the ability to continuously learn (see CONTINUAL-LEARNING section, below,) [3]. The authors list their AI-components as: [DL, ANN, CV, R/IA, NLP, SR/S, ES, KRR, PSO, AR/VR, and SN/GA, acronyms which will be abundantly familiar for those with a masters-level or phd-level education and training in computer-science, but which are listed in the Appendix-1 as reference. The authors also define 'multiple intelligence's as a series of quotients (Q): physical PQ, intelligence IQ, emotional EQ, social SQ, creative CQ, and moral MQ [4].

Both research papers underscore the necessity of meta-learning in the development of AGI by focusing on the adaptability and self-improvement required for intelligent systems to function across complex, evolving domains. Sukhobokov et al. (2024) highlight the importance of meta-learning as part of AGI's core architecture, allowing systems to improve their learning processes over time.

Similarly, Cichocki and Kuleshov (2020) emphasize that continual learning and the integration of multiple intelligence's — which require the system to adapt its methods based on diverse types of intelligence — are essential for AGI. Meta-learning enables AGI to not only perform tasks but to learn how to learn effectively in new and complex environments, making it an indispensable mechanism in the creation of truly autonomous and intelligent systems.

Meta-learning, as highlighted in both sets of research, emerges as a foundational capability for the development of AGI by enabling systems to dynamically optimize their learning strategies in response to complex, real-world challenges. Both Sukhobokov et al. (2024) and Cichocki and Kuleshov (2020) emphasize that for AGI to achieve human-like intelligence, it must be equipped not just with the ability to learn but with the meta-cognitive ability to adapt its learning mechanisms.

Sukhobokov's analysis of cognitive architectures underscores the necessity of incorporating meta-learning as a key component of AGI's self-development and awareness. Meanwhile, Cichocki and Kuleshov's focus on multiple intelligences suggests that meta-learning is critical for AGI to effectively navigate various social, emotional, and ethical contexts. In both cases, meta-learning is not just about task-specific learning but about optimizing the agent's overall approach to learning, ensuring flexibility and long-term adaptability in an ever-changing environment, ultimately bridging the gap between narrow AI and true AGI.

This insight underscores how meta-learning is central to both understanding and advancing AGI, not only as a technical feature but as a crucial mechanism for AGI's development in terms of autonomy, adaptability, and generalization across diverse cognitive functions.

## FEW-SHOT LEARNING

Though focused on the specialized application of characterizing materials with electron micrographs via a gen-AI pipeline to analyze the structures of semiconductor materials, the approach detailed in Srinvias, et al. (2024,) [5] exemplifies, in part, how few-shot-learning is an integral part of generalization as a component of AGI [5].

Few-shot-learning provides LLM's with only a limited number of examples to guide it's output based on the prompt engineering chosen. In the author's work, it's detailed via in-context learning with few-shot-prompting with MultiModal Models (GTV-4V,) (language and vision model,) whereby the limited examples allow analogous learning from prior knowledge (Brown et al. (2020)) [16]. This is done via data-efficient conditional probability distributions -- and this is a critical component on the path to AGI -- allowing for better performance on difficult tasks a potential AGI has yet to be exposed to via labeled data. Few shot learning is best applied for domain-specific problems. To extrapolate from the original research of [5] and connect FSL to the broader development of AGI (Artificial General Intelligence), five key insights can be drawn:

1. *Generalization via Limited Data:* At its core, the power of FSL lies in its ability to generalize from very few examples. In the context of AGI, this is critical. AGI must be capable of learning across diverse domains with minimal supervision or training data. The ability of a model like GPT-4V (a Multi-modal language-vision model) to perform well with limited examples demonstrates how AGI might handle novel, unforeseen tasks without requiring extensive retraining. The model uses prior knowledge to infer new insights, paralleling human cognitive capabilities. (Lake et al. (2017)) [116].

2. *In-context Learning:* This aspect of FSL, where models learn from examples provided in the context of the prompt itself, is integral to how AGI "understands" and adapts to unfamiliar tasks. The model interprets the framework of its pre-existing knowledge, applying it to new problems and scenarios to mimic human generalization (Brown et al., 2020) [116].

3. *Multi-modal Integration:* The integration of both vision and language models, such as the GPT-4V, points to a critical component for AGI: the ability to process and synthesize information across multiple sensory modalities (e.g., vision and language). AGI must integrate diverse forms of input (such as visual, textual, and possibly auditory data) and produce outputs that are informed by a holistic understanding of the world. Few-shot learning in such a Multi-modal context demonstrates how AGI could approach a complex real-world problem—like analyzing materials—by rapidly adapting to new data types and scenarios (Baltrušaitis et al., 2018) [103].

4. *Data-Efficiency and Transfer Learning:* The use of data-efficient conditional probability distributions in FSL models also plays into the future of AGI. The ability to perform well with little data, through transfer learning and by leveraging existing knowledge, allows AGI to overcome challenges in domains where labeled data is scarce or expensive. This is a major advantage in real-world applications, where abundant labeled data is not always available (Pan & Yang, 2010) [117].

5. *Domain-Specific Applications as a Foundation for AGI:* Although FSL is most effective in domain-specific applications, the principles it embodies (rapid learning from few examples, adaptability, and Multi-modal reasoning) are essential stepping stones toward AGI. As AGI evolves, it will likely start with specialized tasks, building up a broad, flexible foundation for cross-domain learning and generalization. FSL's role in AGI development is indispensable, as it underpins AGI's ability to learn efficiently, adapt to new and unseen tasks, and generalize knowledge across different contexts. AGI will require leveraging few-shot techniques in complex, Multi-modal environments to achieve flexibility and cognitive versatility seen in humans. This approach, as highlighted in the work of Srinivas et al. (2020) [5], lays foundational principles for creating a robust, general-purpose system capable of reasoning and decision-making across many domains.

## CONTINUAL LEARNING

Continual learning in automated-intelligence like AI and AGI is the process of acquiring knowledge and transferring it to new domains.

An early entry in pre AGI systems of OpenNARS, OpenCog, and AERA, towards moving from proof-of-concept to scalable-products, was the DFRE framework introduced in Latapie and Kilic (2020) [7], which united ML, DL and probabilistic-programming with AIKR reasoning, with the underlying DFRE theme being 'knowledge is hierarchical-structure,' along with levels of abstraction, with the authors experiment focused on one-shot instead of few-shot learning in the physical world (smart city planning, retail-space shelving, etc…,) with these memory functions acting as data-structures in memory.

The DFRE framework introduced by [7] in OpenNARS, OpenCog, and AERA represents a significant step toward scaling AI systems with real-world applications, highlighting the crucial role of hierarchical knowledge structures and levels of abstraction.

In the context of AGI development, this research is important because it emphasizes one-shot learning in real-world domains, (again, smart city planning and retail shelving,) a crucial feature for AGI systems that need to adapt quickly to new, unseen scenarios with minimal data. The integration of machine learning (ML), deep learning (DL), probabilistic programming, and AIKR reasoning aligns with the direction of AGI by promoting systems that can reason, generalize across domains, and update their knowledge structures dynamically. The memory functions acting as data structures in DFRE reflect a foundational approach to how an AGI system might store and recall information, enabling more efficient learning and adaptability in complex, ever-changing environments.

In "Manhood of Humanity" (1921), Alfred Korzybski [8] argued human cognition and progress are shaped by our capacity for symbolic thought and time-binding via language, a concept central to the development of AGI today.

Korzybski is best known for developing General Semantics, a field of study on how language and symbols influence human perception and behavior. His work in 1921 laid the foundation for understanding how humans interpret the world around them through the lens of language and symbols, and how these interpretations shape thoughts, actions, and societies. LLMs and MultiModal models harness massive linguistic and symbolic data to generalize and evolve across domains, mirroring Korzybski's notion of cognitive evolution and the need for precise semantic on the path to true AGI [8].

The central themes and insights from "Manhood of Humanity" include the human capacity for symbolic thought—primarily through language—as a defining feature of humanity. Unlike animals, humans have the ability to create complex systems of symbols that represent not just objects in the world, but abstract concepts, relationships, and ideas. This capacity allows humans to think about the future, reason, and make predictions. The concept of "Time-Binding" refers to the ability of humans to accumulate and pass on knowledge across generations. In contrast to other animals, humans store and transmit information, which allows for progress, the accumulation of science, culture, and technology [8].

Time-binding, according to Korzybski, is what makes humanity distinct and capable of advancing as a species. This is linked to language as the medium that enables the transmission of knowledge, which in plain terms explains the impact of NLP as the evolutionary root of the current explosion in ai_LLM's.

The Human "Map" vs. the "Territory" emphasized the map and the territory—a metaphor for how language and symbols (the map) represent the reality of the world (the territory), however humans confuse the two, thinking words and symbols we use to represent the world are the world itself. This leads to misunderstandings and conflicts. And transferring this transcendence to AGI can be achieved via the paradigm of continual-learning [8].

Korzybski even pointed to the future relevance of ethics in AI and AGI, as he explored the idea that humans, due to their capacity for abstract thinking and time-binding, have a unique responsibility to use their cognitive powers wisely [8]. He viewed this responsibility as a moral imperative, suggesting that human beings must evolve their ways of thinking and communicating to ensure the survival and prosperity of humanity. This early thought framework for ethical AI and AGI should act as scriptural guidance in the arms-race to perfect AI and create AGI.

Korzybski's work in his 1921 book set the stage for the later development of General Semantics, which has influenced psychology, linguistics, communications, and the development of artificial intelligence and cognitive science [8]. His ideas remain relevant today, particularly in how language shapes cognition and perception. This early example of continual learning remains relevant for current formulas and architectures on the path to true AGI.

**G_enhanced / (e_Generalization)** mathematical notation:.

**Key Concepts:**

- Meta-Learning (ML) enhances Generalization (G) by improving learning processes over time.
- Few-Shot Learning (FSL) enables Generalization (G) from limited examples.
- Continual Learning (CL) facilitates knowledge transfer and adaptation, contributing to Generalization (G).
- Multiple Intelligences (MI) and their quotients (Q) impact Generalization (G).
- Hierarchical Knowledge Structures (HKS) and Time-Binding (TB) support Continual Learning (CL), which supports Generalization (G).
- Generalization is inversely related to e_Generalization (ε).

Notation:

✓ Inverse Relationship:

✓ Meta-Learning Influence:

- Let ML represent Meta-Learning.
- Formula: **ε = f(1/ML)** (Error is a function of the inverse of Meta-Learning, meaning higher Meta-Learning reduces error).

✓ Few-Shot Learning Influence:

- Let FSL represent Few-Shot Learning.
- Formula: **ε = g(1/FSL)** (Error is a function of the inverse of Few-Shot Learning, meaning higher Few-Shot Learning reduces error).

✓ Continual Learning Influence:

- Let CL represent Continual Learning.
- Formula**: ε = h(1/CL)** (Error is a function of the inverse of Continual Learning, meaning higher Continual Learning reduces error).

✓ Multiple Intelligences Influence:

- Let MI represent Multiple Intelligences, Let Q represent Quotients.
- Formula: **ε = j(1/MI)** and **MI = k(Q)** (Error is a function of the inverse of Multiple Intelligences, Multiple Intelligences are function of Quotients).

✓ Hierarchical Knowledge Structures and Time-Binding Influence:

- Let HKS represent Hierarchical Knowledge Structures, Let TB represent Time-Binding.
- **Formula: CL = l(HKS, TB)** and **ε = h(1/CL)** (Continual Learning is function of Hierarchical Knowledge Structures and Time-Binding, which impacts error).

Consolidated Formula for e_Generalization (ε)**:**

- **ε = F ( 1/ML, 1/FSL, 1/CL, 1/MI, Q, HKS, TB )**

e_Generalization (ε) is a function (F) of the inverse of Meta-Learning, Few-Shot Learning, Continual Learning, and Multiple Intelligences, as well as the Quotients, Hierarchical Knowledge Structures, and Time-Binding.

This formula shows that increasing these learning capabilities reduces the error in generalization.

- **ε = F ( 1/ML, 1/FSL, 1/CL, 1/MI, Q, HKS, TB )**

Integration:

Introduce Inverse Proportionality within Code-Notation Structure:

**ε = 1 / ( 1/ML' + 1/FSL' + 1/CL' + 1/MI ')**

Where ML', FSL', CL', and MI' represent the *effective* learning capabilities, possibly modified by other factors.

Explicitly Include Functions for MI and CL:

**ε = 1 / (1/ML' + 1/FSL' + 1/g(HKS, TB) + 1/f(Q))**

Acknowledge Other Influences by introducing a Modifying Function "h"

**ε = h (1 / (1/ML' + 1/FSL' + 1/g (HKS, TB) + 1/f (Q) )**, OtherFactors) (OtherFactors = raw values: Q, HKS or TB, or other influence on error.)

Simplify and Generalize:

**ε = h(1 / Σ(1/LearningCapabilities), OtherFactors),** Where LearningCapabilities **= { ML', FSL', g(HKS, TB), f(Q) }**

Implied Pseudocode (next section) :

- **e_Generalization = 1 / (MetaLearning + FewShotLearning + ContinualLearning + MultipleIntelligences)**
- **MultipleIntelligences = Function(Quotients)**
- **ContinualLearning = Function ( HierarchicalKnowledgeStructures, TimeBinding )**

Notation:

Inverse Relationship and Sum of Learning Components:

**ε = 1 / ( ML + FSL + CL + MI )**

**Multiple Intelligences as a Function of Quotients:**

**MI = f (Q)**

Continual Learning as a Function of Hierarchical Knowledge Structures and Time Binding:

**CL = g (HKS, TB)**

Consolidated Formula:

**ε = 1 / ( ML + FSL + g (HKS, TB) + f(Q) )**

**Where:**

- ε = e_Generalization (error in generalization)
- ML = Meta-Learning // `meta_learning()`
- FSL = Few-Shot Learning // `few_shot_learning()`
- CL = Continual Learning // `continual_learning()`
- MI = Multiple Intelligences // `define_multiple_intelligence_metrics()`
- Q = Quotients // `load_reviewed_cognitive_architectures() [ETC…]`
- HKS = Hierarchical Knowledge Structures // `update_knowledge_graph()`
- TB = Time-Binding // `create_time_binding_structure()`
- f() and g() are functions. // `function meta_learning()[ETC..], function generate_tensors()`

Pseudo-code for component_1 ( **G_enhanced / (e_Generalization)**

(structured pipeline implemented in TensorFlow input to component_5 **CV_Adapt.)**

```
# Pseudo-Code for G_enhanced Tensor Generation

# Step 1: Define Meta-Learning Components
function meta_learning():
    architectures = load_reviewed_cognitive_architectures()
    filtered_architectures = filter_for_AGI_capabilities(architectures)

    key_components = {
        "consciousness":
extract_architectures_with_consciousness(filtered_architectures),
        "subconscious":
extract_architectures_with_subconscious(filtered_architectures),
        "worldview": extract_architectures_with_worldview(filtered_architectures),
        "reflection": extract_architectures_with_reflection(filtered_architectures),
        "multiple_intelligences": define_multiple_intelligence_metrics()
    }

    return key_components

# Step 2: Define Few-Shot Learning Mechanism
function few_shot_learning(input_data):
    model = load_pretrained_MultiModal_Model()
    few_shot_examples = prepare_few_shot_prompts(input_data)
    predictions = model.predict(few_shot_examples)

    generalization_metrics = evaluate_generalization(predictions)
    return generalization_metrics

# Step 3: Implement Continual Learning Strategy
function continual_learning():
    memory_structure = initialize_memory_hierarchy()
    knowledge_graph = build_dynamic_knowledge_graph()

    while system_running():
        new_data = fetch_real_world_data()
        updated_memory = update_memory(memory_structure, new_data)
        update_knowledge_graph(knowledge_graph, updated_memory)

    return knowledge_graph
```

```
# Step 4: Symbolic Representation for AGI
function symbolic_representation():
    language_processing = initialize_NLP_model()
    symbolic_memory = create_time_binding_structure()
    for input_text in continuous_data_stream():
        semantic_analysis = process_semantics(language_processing, input_text)
        update_symbolic_memory(symbolic_memory, semantic_analysis)

    return symbolic_memory
```

```
# Step 5: Generate Tensor Representations for TF Model
function generate_tensors():
    meta_features = meta_learning()
    few_shot_features = few_shot_learning(meta_features)
    continual_learning_data = continual_learning()
    symbolic_data = symbolic_representation()

    tensor_inputs = convert_to_tensor([meta_features, few_shot_features,
continual_learning_data, symbolic_data])
    return tensor_inputs
```

```
# Execution
tensor_data = generate_tensors()
```

**G_enhanced / (e_Generalization)** summary:

- **Meta-Learning (ML)** enhances Generalization (G) by improving learning processes over time.
- **Few-Shot Learning (FSL**) enables Generalization (G) from limited examples.
- **Continual Learning (CL)** facilitates knowledge transfer and adaptation, contributing to Generalization (G).
- **Multiple Intelligences (MI)** and their quotients (Q) impact Generalization (G).
- **Hierarchical Knowledge Structures (HKS)** and **Time-Binding (TB)** support Continual Learning (CL), which supports Generalization (G).
- Generalization is inversely related to e_Generalization (ε).
- **final output of this pseudo-code (once source-coded,) is adapted into the TensorFlow model to generate tensors, used as input to component_5 CV_adapt.**

*"Artificial intelligence is the study of agents that receive precepts from the environment and take actions to maximize their chances of success." Stuart Russell & Peter Norvig (1995)* [72] .

NEURO-SYMBOLIC AI

Berlot-Atwell (2021) [9] reviewed Neuro-symbolic approaches to visual-questioning-answering via models of AI and ML as an aspect towards AGI [9], where natural-growth, traceability, transfer-learning, few-shot-learning, and self-awareness of limitations were found in neuro-symbolism within performance, scalability and training-data practical considerations [9].

Hamilton et al. (2024) [10], in part, seemed to advocate for the approach, in part, taken by Sukhobokov, et al, (2024) [2]. with respect to the issue of a more methodical approach of human reasoning and appropriate benchmarks [10], extrapolated in the case of [2], as potential AGI architectures.

More importantly as detailed in [10] which reviews studies of neuro-symbolism for the NLP-component of AI-LLM's,) [10] (itself 1 of 5 components of this papers present formula for AGI,) systems where logic is compiled into the neural-network led to the highest scoring goal completion, while surprisingly, the *type* of neural-network did not correlate with the highest scoring goal completion [10]. And Hagos, D.H. and Rawat, D.B. (2024)[12] explore Neuro-symbolic AI for military and civilian applications through the lens of ethics, strategy and technology to automate complex intelligence analysis and strengthen autonomous systems [12]. Via improved threat-detection, optimized logistics, and enhanced situational awareness [12] the strength of pairing neural-networks and symbolic-reasoning is detailed in the context of AI.

Each of these contribute to alignment with the evolving understanding of Neuro-symbolic AI (NS-AI) as a viable component of AGI. Berlot-Atwell (2021) [9] provides foundational understanding of Neuro-symbolic approaches in visual question answering (VQA) via integration of symbolic reasoning with neural networks which in turn enable traceability, scalability, and adaptability in AI systems [9]. The importance of self-awareness of limitations in AI systems also plays a crucial role in making AI more robust and transparent, characteristics vital for AGI. This directly supports the viability of Neuro-symbolic AI for AGI by emphasizing its ability to scale, reason symbolically, and improve incrementally.

Hamilton et al. (2024) underscores the importance of methodical human reasoning in the development of AGI, and Hagos & Rawat (2024) explores the application of neuro-symbolic AI in high-stakes military and civilian sectors, in intelligence analysis and autonomous systems, with these applications showcasing neuro-symbolic AI be as a proxy for sophisticated ethically-informed autonomous systems, which is essential for AGI.

These studies support the thesis neuro-symbolic AI represents a partial component toward achieving AGI, because pattern recognition strengths of neural networks and logical interpretative capabilities of symbolic reasoning are better at generalization, learning efficiency, and autonomous decision-making, all critical for AGI.

REINFORCEMENT-LEARNING

Reinforcement learning from human-feedback (RLHF) is a technique which aligns LLM's with human preferences, and is a major key to the success of major ai chat-bots such as chatGPT.

However, in Alexander (2020,) [14] the reward-based learning model is challenged by pointing out that AGI requires more than just reinforcement-learning (RL,) and or reinforcement-learning from human-feedback (RLHF) [14], due to the limits of reward-maximization, which performs well in structured-environments, but degrades in dynamic-environments.

Alexander offers lemmas via his Archimedean Trap (reward-optimization-limitation, and, context-specificity of learning,) [14] which reveal the inherent limits in the scope of intelligence an agent can exhibit, and by default, pushes for greater robustness via: causal-reasoning, symbolic processing, contextual adaptation, and learning from sparse or even zero rewards -- and each of these components are supported and presented within the ai-2-agi architecture presented within the main body of *this* paper.

And where RL / RLHF would comprise, in very rough terms, approximately 7% of AI, or in the least point to it's importance for fine-tuning models [16], (Stiennon et al. (2022)) [17] and, approximately 18% of AGI, or in the least point to it's increased importance for the quantitative-role (of RL / RLHF,) within AGI [19] [20], regardless of the percentage-composition, RL / RLHF is one of two integral components for enhanced_Reasoning (e_Reasoning), itself a component of AGI.

And with respect to two common issues of everyday life, forgetting, and math computations, overcoming these common issue obstacles is a substantial undertaking for AI and AGI within the context of e_Reasoning.

Rusu et. al. (2022) [20] addresses the issue of forgetting, via the progressive-networks [20] approach, which the authors state is immune to forgetting, tested via RL tasks such as Atari and three-dimensional mazes. And progressive-networks also applies to the continual-learning sub-component of e_Generalization, as discussed previously in that section: "where agents not only learn (and remember) a series of tasks experienced in sequence, but also have the ability to transfer knowledge from previous tasks to improve convergence speed." [20].

Du et. al. (2023) [22] addresses the issue of math, (thus far a challenge for most LLM's focused mostly on text and language,) via the society-of-minds [22] approach via multiple language model instances which propose and debate outputs to ultimately generate a common final answer, which clearly advances LLM capabilities.

**R_enhanced / (e_Reasoning)** math-notation

**Key Concepts:**

- R_enhanced (Enhanced Reasoning) is inversely proportional to e_Reasoning (Error in Reasoning).
- Neuro-Symbolic AI (NS-AI) enhances Reasoning (R).
- Reinforcement Learning (RL/RLHF) enhances Reasoning (R), but has limitations.
- Forgetting and Mathematical Computation are challenges for Reasoning (R).
- Progressive Networks (PN) address Forgetting.
- Society of Minds (SM) addresses Mathematical Computation.

**Notation:**

✓ Inverse Relationship:

- Let R represent R_enhanced. Let ε_r represent e_Reasoning.
- Formula: $\mathbf{R \propto 1/\varepsilon\_r}$

✓ Neuro-Symbolic AI Influence:

- Let NS represent NS-AI.
- Formula**: ε_r = f(1/NS)** (Error in reasoning decreases with increased NS-AI).

✓ Reinforcement Learning Influence:

- Let RL represent RL/RLHF.
- Formula: **ε_r = g(1/RL)** (Error in reasoning decreases with increased RL, but with limitations).
- Limitation: RL is context-specific and limited by reward maximization.

✓ Addressing Forgetting:

- Let PN represent Progressive Networks.
- Formula: **ε_r(forgetting) = h(1/PN)** (Forgetting-related error decreases with PN).

✓ Addressing Mathematical Computation:

- Let SM represent Society of Minds.
- Formula: **ε_r(math) = j(1/SM)** (Math-related error decreases with SM).

✓ Combined Influence:

- **ε_r = k(1/NS, 1/RL, 1/PN, 1/SM)** (Error in reasoning is a function of NS-AI, RL, PN, and SM).

✓ NeuroSymbolic AI and Human Reasoning

- **NS -> human reasoning.**

Consolidated Conceptual Formula for e_Reasoning (ε_r):

- **ε_r = k(1/NS, 1/RL, 1/h(PN), 1/j(SM))**

Start:

- **ε_r = k (1/NS, 1/RL, 1/h(PN), 1 / j(SM) )**

Function Definitions:

- **NS(x)** = Neuro-Symbolic output for input x.
- **RL(x, r)** = Reinforcement Learning output for input x and reward r.
- **CR(x)** = Causal Reasoning output for input x.
- **CA(x)** = Contextual Adaptation output for input x.
- **PN(x)** = Progressive Networks output for input x.
- **SM(x)** = Society of Minds output for input x.

Functional Mapping and Sequential Structure:

- Assume **k (a, b, c, d) = 1 / F ( a', b', c', d' )**
- Therefore, **ε_r = 1 / F ( NS', RL', PN', SM ')**

Explicit Function Transformations using Defined Functions:

- **NS'** = NS(input_data)
- **RL'** = RL(NS(input_data), r)
- **PN'** = PN(CA(CR(RL(NS(input_data), r))))
- **SM'** = SM(PN(CA(CR(RL(NS(input_data), r)))))

Substitution:

- **ε_r = 1 / F ( NS(input_data ), RL ( NS( input_data ), r), PN(CA(CR(RL(NS(input_data), r)))), SM(PN(CA(CR(RL(NS(input_data), r))))))**

Define F as a Sequential Composition / Apply F to the Substituted Transformations:

- **Let a** = NS(input_data)
- **Let b** = RL(NS(input_data), r)
- **Let c** = PN(CA(CR(RL(NS(input_data), r))))
- **Let d** = SM(PN(CA(CR(RL(NS(input_data), r)))))
- Then, **F(NS(input_data), RL(NS(input_data), r), PN(CA(CR(RL(NS(input_data), r)))), SM(PN(CA(CR(RL(NS(input_data), r))))))** = SM(PN(CA(CR(RL(NS(input_data), r)))))
- Therefore, **ε_r = 1 / (SM ( PN( CA ( CR ( RL ( NS (input_data), r))))))**

Key Operations and Components:

- **Neuro-Symbolic Integration (NS):** Combines neural network output with symbolic reasoning.
- **Reinforcement Learning (RL):** Optimizes output based on human feedback and rewards.
- **Causal Reasoning (CR):** Derives cause-effect relationships.
- **Contextual Adaptation (CA):** Adjusts reasoning based on context.
- **Progressive Networks (PN):** Handles forgetting.
- **Society of Minds (SM):** Solves math problems.
- **e_Reasoning (ε_r):** The overall reasoning error.

Mathematical Notation:

Reward Computation:

- **r = reward(human_feedback)**

Sequential Application of Functions:

- **ε_r = 1 / ( SM (PN (CA (CR (RL (NS (input_data), r ))))))**

Expanded notation

- **ε_r = 1 / (SM(PN(CA(CR(RL(NS(input_data), reward(human_feedback)))))))**

**Where:**

- ε_r = e_Reasoning (error in reasoning) // `apply_symbolic_reasoning()`
- R = R_enhanced (enhanced reasoning) // `process_neuro_symbolic_integration()`
- NS = Neuro-Symbolic AI // `initialize_neuro_symbolic_model()`
- RL = Reinforcement Learning/RLHF // `initialize_reinforcement_learning_system()`
- PN = Progressive Networks // `apply_progressive_networks()`
- SM = Society of Minds // `solve_math_problems(), run_multiple_models()`
- h() and j() and k() functions // `initialize_neuro_symbolic_model(), solve_math_problems()`

The formula represents inverse between e_Reasoning (ε_r) and the sequential application of reasoning functions. The input data is passed through each function in a specific order: Neuro-Symbolic Integration, Reinforcement Learning (with reward computation), Causal Reasoning, Contextual Adaptation, Progressive Networks, and Society of Minds. Final result of the series of functions is inverted, to represent error rate. The reward is a function of the human feedback.

Pseudo-code for component_2 ( **R_enhanced / (e_Reasoning** )

(structured pipeline implemented in TensorFlow input to component_5 **CV_Adapt.)**

```
# Pseudo-code for Enhanced Reasoning (e_Reasoning) Component of AGI

# Initialization: Setup model parameters
initialize_neuro_symbolic_model()  # Integrate symbolic reasoning with neural
network
initialize_reinforcement_learning_system()  # Setup RL system with Human feedback
(RLHF)

# Handling Neuro-Symbolic Integration
def process_neuro_symbolic_integration(input_data):
    # Combine neural-network capabilities with symbolic reasoning
    neural_network_output = run_neural_network(input_data)
    symbolic_reasoning_output = apply_symbolic_reasoning(neural_network_output)

    # Return integrated output (traced, scalable, adaptable)
    return symbolic_reasoning_output

# Reinforcement Learning (RL) Component
def process_reinforcement_learning(input_data):
    # Apply RLHF to adapt model to human preferences
    human_feedback = gather_human_feedback(input_data)
    reward = compute_reward(human_feedback)

    # Apply reward optimization based on RL principles
    optimized_output = optimize_output_with_reward(input_data, reward)

    # Return the optimized output
    return optimized_output

# Causal Reasoning & Contextual Adaptation
def apply_causal_reasoning(input_data):
    # Perform causal reasoning to understand the cause and effect in the given context
    causal_output = perform_causal_reasoning(input_data)
    return causal_output

def apply_contextual_adaptation(input_data):
    # Adjust reasoning based on changing contexts and environments
    adapted_output = adapt_to_context(input_data)
    return adapted_output
```

```
# Handling Forgetting Issue (via Progressive Networks)
def handle_forgetting(input_data):
    # Use progressive networks to handle forgetting, ensuring continual learning
    progressive_network_output = apply_progressive_networks(input_data)
    return progressive_network_output

# Math Problem Solving (via Society of Minds Approach)
def solve_math_problems(input_data):
    # Use multiple model instances to collaboratively solve math problems
    proposed_answers = run_multiple_models(input_data)
    final_answer = resolve_conflicts(proposed_answers)
    return final_answer

# Main e_Reasoning Function
def e_reasoning(input_data):
    # Step 1: Process Neuro-Symbolic Integration
    symbolic_output = process_neuro_symbolic_integration(input_data)

    # Step 2: Apply Reinforcement Learning for human feedback optimization
    rl_output = process_reinforcement_learning(symbolic_output)

    # Step 3: Apply causal reasoning
    causal_output = apply_causal_reasoning(rl_output)

    # Step 4: Adapt to context changes
    contextual_output = apply_contextual_adaptation(causal_output)

    # Step 5: Handle forgetting issue with Progressive Networks
    final_output = handle_forgetting(contextual_output)

    # Step 6: Solve math problems (if applicable)
    math_output = solve_math_problems(final_output)

    # Return final output (tensor form for further processing)
    return math_output

# Final Step: Generate Tensor for input to next module
tensor_output = e_reasoning(input_data)  generate_tensor_from_output(tensor_output)
```

**R_enhanced / (e_Reasoning)** summary:

- **Neuro-Symbolic Integration**: Combines neural networks with symbolic reasoning.
- **Reinforcement Learning (RLHF)**: Adapts the model to human preferences through feedback.
- **Causal Reasoning**: Understands cause and effect within the input data.
- **Contextual Adaptation**: Adjusts the model's reasoning based on the context.
- **Forgetting via Progressive Networks**: Prevents forgetting and enables continual learning.
- **Math Problem Solving via Society of Minds**: Uses multiple models to solve complex problems and generate consensus.
- **final output of this pseudo-code (once hard-code source-coded,) is adapted into TensorFlow to generate tensors, used as input to component_5 CV_adapt.**

## ai_LLMs

*"The ability of a machine to improve its performance through experience and feedback is one of the hallmarks of intelligent behavior." (Arthur Samuel, 1959)* [80].

According to Francois Chollet, creator of ARC-AGI (the only formal benchmark of AGI progress,) and Keras, “LLMs are trained on unimaginably vast amounts of data, yet remain unable to adapt to simple problems they haven’t been trained on, or make novel inventions, no matter how basic.” (Chollet, (2023)) [23].

And some researchers and practitioners believe solving ARC (Chollet’s unbeaten 2019 Abstraction and Reasoning Corpus for Artificial General Intelligence) (which measures abstract pattern recognition similar to an IQ test,) is how AGI emerges.

According to Mumuni (Mumuni et al. (2025) [24], “generic LLMs are severely limited in their generalist capabilities” [24] and embodiment, causality and memory must be addressed to achieve human-level general intelligence, which supports the thesis that ai-LLMs will be a component of a future true AGI, but not the main component and not the only component” [24].

The authors focus is on the MultiModal LLMs which have renewed interested in the journey to true AGI, and they do so by presenting fundamentals of cognition which support true AGI, (in their definition, “AI systems that exhibit broad intellectual abilities and are able to perform high-level cognitive tasks…”) such as perception, self-awareness, reasoning and planning [25] via state-of-the-art techniques [25]. The authors then differentiate the scale of AI, AGI and ASI, with ASI being super-omniscient.

Expanding on the authors definition of AGI, AGI solves sensorimotor control, perception, context understanding and common and analytical reasoning [25], and relate it closer to biological intelligence, (which again supports the thesis of *this* paper as it relates to component_5 DTB (see that section,) via autonomous and goal driven actions, retaining and accumulating relevant information in memory, and performance of high level abstract and commonsense reasoning [25].

So in this sense, the more concise and tested the definition of AGI is elicited, the greater the disparity between AI / ai_LLMs and AGI comes into focus, and concurrently, the greater the relevance of including AI / ai_LLMs as an integral component of AGI also comes into focus.

So the key behind the importance of including AI / ai_LLMs as a component of AGI, is their ability to build and train massive NN’s on multi modal data.

And the key behind why AI / ai_LLMs will *not* be the main and the only technological component of AGI is: their understanding of context is limited due to their digital construct of observed patterns, and lack of sentience. However, Mumuni et al. Make the case that AI / ai_LLMs can actually be leveraged to achieve true AGI, though, to be clear, that is *not* the position of *this* paper, and as stated, AI, ai_LLMs are approximately $1/5^{th}$ the overall functional equation behind the architecture of true AGI as presented.

The limited contextual understanding and lack of sentience inherent in AI and ai_LLMs, due to their digital construct of observed patterns, challenge their role as the sole technological component of AGI. While Mumuni et al. (2025) argue for leveraging AI and ai_LLMs to achieve AGI, this paper posits that they represent approximately one-fifth of the functional equation underlying AGI architecture. The pursuit of AGI can be likened to Zeno's Paradox of infinite, where a person takes an infinite number of half-the-distance-to-the-wall steps but never quite makes all the way to actually touching the wall with their foot. So too with AGI, despite significant advancements, particularly with ai_LLMs, the remaining steps toward AGI remain unquantifiable. .

The 2023 thesis on emergent abilities in ai_LLMs scaling up to improve capabilities (Xie et al. (2023) [32] was recently challenged by the emergence of DeepSeek AI, (Rahman et al. (2025)) [118] (DeepSeek implements an MoE which triggers only the most relevant parameters of the model, versus ChatGPT dense-transformer +RHLF, versus Gemini multimodal-transformer,) which demonstrated emergent abilities at a smaller scale. This development supports he notion that model scale may not be the sole determinant of emergent abilities, a concept with significant economic implications. This underscores the intensity of the global pursuit of AI and AGI (Sutton, 2019) (Sutton, 2023)[118.] Given the extensive research and development in e_Generalization, e_Rationalization, and ai_LLMs, this paper primarily focuses on the DTB and CV_Adapt components. The latter components have received comparatively less attention in published research, particularly concerning the approach of using computer-vision as a central-processor for the of the AGI to ‘read’ the inputs from the other components, the proposed integration of DTB, and the holistic approach to combining all five technologies into a cohesive ai2agi framework to ultimately create a possible blueprint for AGI-dna.

This preliminary literature review supports the thesis that while ai_LLMs are a significant technological advancement, they are not the sole component of AGI. Instead, they will be an integral part of a multi-component AGI architecture, as outlined in this research

That said, ai_LLMs will, at the same time, be an integral component of true AGI, and their importance cannot be overstated.

Ai_LLM's

Key Concepts:

- **ai_LLMs: Large Language Models**
- **AGI: Artificial General Intelligence**
- **ARC: Abstraction and Reasoning Corpus**

Mathematical Notation:

- ✓ Limitations of ai_LLMs:
  - **L(ai_LLMs) ≠ AGI**
  - **ARC → AGI (ARC solves implies AGI)**
- ✓ ai_LLMs as a Component:
  - **AGI = f(ai_LLMs, [four] Other Factors)**
- ✓ Core AGI Functions:
  - **AGI = g ( Perception, Reasoning, Planning, Context)**
- ✓ Scale Hierarchy:
  - **ai_LLMs < AGI < ASI**
- ✓ Context and Sentience Limits:
  - **L(ai_LLMs) = h(Context Limit, Sentience Limit)**

Start:

- **L(ai_LLMs) = h(Context Limit, Sentience Limit)**
- **AGI = f(ai_LLMs, [four] Other Factors)**
- **ai_LLMs < AGI < ASI**

Introduce Functional Composition:

- **ai_LLM(x) = F ( MM(x), K(x), M(x), L(x), A(x) )**

Incorporate Sequential Dependency:

- **ai_LLM (x) = A ( L ( M ( K ( MM (x) ) ) ) )**

Acknowledge Limitation Function:

- **L(x) = h(Context Limit, Sentience Limit)**

Connect to AGI Function:

- **AGI = f ( ai_LLM(x), [four] Other Factors )**

Hierarchy Remains:

- **ai_LLMs < AGI < ASI**

Mathematical Notation:

Individual Component Functions:

- **MM(x)** = Multimodal Output
- **K(x)** = Context-Aware Knowledge
- **M(x)** = Memory-Enhanced Output
- **L(x)** = Adapted Output
- **A(x)** = Augmented AGI Output

Sequential Application:

- **ai_LLM(x) = A (L (M (K (MM (x) )))))**

Expanded Notation:

- **ai_LLM(input_data) = A (L (M (K (MM (input_data ) )))))**

Where:

- **MM(x):** Process Multimodal Data // `process_multimodal_data()`
- **K(x):** Process Knowledge and Context // `process_knowledge()`
- **M(x):** Apply Memory Integration // `apply_memory_integration()`
- **L(x):** Address Limitations // `analyze_limitations()`
- **A(x):** Augment AGI Capabilities // `apply_agi_augmentation()`
- **ai_LLM(x):** Main ai_LLM Processing // `ai_llm_for_agi()`

Pseudo-code for component_3 ( **ai_LLM's** )

(structured pipeline implemented in TensorFlow, includes handling MultiModal data, addressing limitations, and, integrating LLMs for component_3 of true AGI)

```
# Pseudo-code for AI_LLM Integration in AGI

# Initialization: Setup AI/LLM model parameters
initialize_ai_llm_model()  # Initialize the generic LLM with MultiModal capabilities
initialize_context_understanding()  # Setup context-awareness mechanisms
initialize_perception_system()  # Setup the perception system to process MultiModal
data
initialize_memory_system()  # Initialize memory module for long-term learning

# AI/LLM Key Functionality - Multi-modal Data Handling
def process_multimodal_data(input_data):
    # Process input data from various modalities (text, image, audio, etc.)
    multimodal_output = integrate_multimodal_data(input_data)
    return multimodal_output

# AI/LLM Knowledge and Context Processing
def process_knowledge(input_data):
    # Process raw data through the AI model to extract abstract patterns and knowledge
    knowledge_output = run_ai_llm(input_data)

    # Context-awareness adjustment
    context_aware_output = adjust_for_context(knowledge_output)
    return context_aware_output

# Addressing the Limitations of AI/LLM
def apply_limitations(input_data):
    # Identify limitations in the model (lack of sentience, inability to adapt to novel
problems)
    identified_limitations = analyze_limitations(input_data)

    # Adaptation layer to improve model capabilities or request external processing
(AGI components)
    adapted_output = improve_through_agency(input_data, identified_limitations)
    return adapted_output

# Memory Integration - Enhancing Generalization
def apply_memory_integration(input_data):
    # Apply the memory system to accumulate and retain relevant information
    memory_output = integrate_memory(input_data)
        # Use the memory to enhance generalization and decision-making capabilities
    enhanced_output = use_memory_for_generalization(memory_output)
    return enhanced_output
```

```
# Reinforcement of AGI Components with AI/LLMs
def apply_agi_augmentation(input_data):
    # Reinforce AGI components (e.g., symbolic reasoning, decision-making, etc.) using
AI/LLM
    augmented_output = enhance_with_llm(input_data)
    return augmented_output
```

```
# Main AI/LLM Processing Function for AGI Integration
def ai_llm_for_agi(input_data):
    # Step 1: Process multimodal data through AI/LLM
    multimodal_data = process_multimodal_data(input_data)

    # Step 2: Extract knowledge and adjust for context
    context_aware_knowledge = process_knowledge(multimodal_data)

    # Step 3: Apply memory integration to improve generalization
    memory_enhanced_output = apply_memory_integration(context_aware_knowledge)

    # Step 4: Address limitations of AI/LLM and adapt to improve capabilities
    final_output = apply_limitations(memory_enhanced_output)

    # Step 5: Reinforce AGI capabilities using AI/LLM
    agi_output = apply_agi_augmentation(final_output)

    # Return the final integrated output (tensor form for further processing)
    return agi_output
```

```
# Final Step: Generate Tensor for input to other AGI components
tensor_output = ai_llm_for_agi(input_data)
generate_tensor_from_output(tensor_output)
```

**ai_LLM** summary:

- **Multi-modal Data Handling**: Integrating different types of data (text, image, etc.) into the LLM system.
- **Context Understanding**: Adjusting for context-awareness to improve model reasoning.
- **Limitations Handling**: Recognizing and addressing the limitations of LLMs, such as lack of sentience or inability to handle novel problems.
- **Memory Integration**: Using memory to enhance the LLM's generalization and decision-making capabilities.
- **AGI Augmentation**: Augmenting other AGI components with AI/LLM capabilities to strengthen the overall system.
- **The final output of this pseudo-code (once hard-coded source-coded,) is adapted into TensorFlow to generate tensors, used as input to component_5 CV_adapt.**

## DTB

*"We suggest that any system of ideas, including the nervous system, can be represented by a set of simple logical operations." (McCulloch & Pitts, 1943)* [71].

With the literature reviews provided for the first three components (G_enhanced, R_enhanced, ai_LLM,) (the three components well within the public domain,) the focus now is on the lesser examined, final two components: DTB (this section,) and, CV_Adapt (next section.) To reiterate, as of the date of this project paper there's an abundance of extremely-well researched, financed, published and implemented projects and data on components_1, _2, and _3, (G_enhanced, R_enhanced, ai_LLM.)

So greater attention and focus for this research project is on DTB in conjunction with CV_Adapt (integrated with G_enhanced, R_enhanced, and ai_LLMs,) as the differentiator.

DTB is instantiated via quantum spiking (excitory, inhibitory) neural-circuit models of single neuronal spikes with simulations of billions of neurons, and *trillions* of synapses, accomplished via GPU / DLA accelerated supercomputers. (*See pseudocode in two-sections which follows computational-notation.*)

State of the art hardware for this type of super-computing is why Nvidia commands a ~$2.6 Trillion market cap as of the date of this research project (Nvidia, 2025) [120]. But even with quantum computing, DLA's, and 50 to 100 peta-flop chips from Nvidia and Cerebras, the immense complexity of replicating brain functions presents obstacles with known unknowns, and, unknown unknowns. So breaking down the digital twin approach into sub components, each of which must be solved on its own merit, is as close as state-of-the art can get at this time.

(Shang et al. (2025)) [39] proposes a neuromorphic computing model based on spiking-neural-networks to emulate the way the human brain processes information [39]. The authors architecture includes extraction of brain signals, spike encoding, brain-signal-id, and analysis results.

(Li et al. (2025)) [40] introduced their Behavior-Chain [40] benchmark for evaluating LLM's ability to simulate 15,846 continuous human actions across 1001 unique personas, and concluded even state-of-the-art models have difficulty emulating human behaviour.

Though a different approach to a different application, (Hakim et al. (2025)) [41] encountered challenges (and other successes,) with extraction and integration of symbolic rules in complex environments. The authors ANSR-DT framework for digital twins is a pattern-recognition algorithm for real-time learning and adaptive intelligence [41]. Their CNN-LSTM network with symbolic reasoning achieved a 23% increase in precision for dynamic pattern-recognition.

The rigorous approach in (Qi et al. (2024)) [42] explored the dynamics of a mean field model of the brain, focusing on strength of synaptic connections and signal processing. Their digital-twin-brain computational model includes 378 areas designed by excitory and inhibitory neuronal-nets with incoming synaptic connections proportional to data in a DWI matrix (itself providing information about structural connectivity) [42].

Using linear-algebra, differential-equations, probability-theory, and graph-theory, the authors developed a second-order (average firing-rate, *and*, variability in the neuron firing,) dynamic (simulating brain changes over time,) mean field model (average activity of large groups of neurons) [42] to bridge the gap between individual neurons and the overall dynamics of the brain.

But the development of a Digital Twin Brain (DTB) within AGI raises ethical questions (**Appendix 5**.) So while the concept of digital consciousness prompts discussions about potential rights, it's also important to acknowledge the practical context. If AGI achieves consciousness, it likely signifies a level of advancement that would unlock solutions to critical human challenges such as poverty, disease, and resource scarcity. So, ethical considerations should focus on ensuring AGI's development is aligned with maximizing human well-being and addressing pressing needs. Simultaneously, its important to remain mindful of implications of digital consciousness and establish safeguards to prevent unintended consequences. With that brief ethical primer, we proceed.

Each of these, and other established and recent projects on digital twins highlight a segment or approach which when taken as a whole might point to the importance of including DTB as a component of AGI.

A possible outline for the DTB component is presented, below.

For perspective, we revisit the formula:

**AGI ≈ f( G_enhanced , R_enhanced, LLM, DTB, CV_Adapt )**

where:

G_enhanced = enhanced-Generalization (meta & few-shot & continual learning)

R_enhanced = enhanced-Reasoning (neuro-symbolic AI, reinforcement-learning w/ environmental-interaction).

ai_LLM = Understanding (ai-LLMs)

DTB = Digital Twin Brain (representations of Consciousness/Self-Awareness)

CV_adapt = Adaptation (Computer Vision analyzing tensor-representations of: G_enhanced, R_enhanced, ai_LLM, & DTB.)

DTB = Digital Twin (representations of Consciousness/Self-Awareness)

So, to start DTB, we first review [48][49][50][51][52] ***just the electrical activity of a single neuron:***:

(We'll add chemical, glial cell, brain-region, connectome, dynamic-activity, and internal factors to this single neuron, and then scale the single neuron to billions of neurons, and then scale the billions of neurons to *trillions* of synapses as this schematic unfolds in the next ~10 pages.)
To start, [100][107][112][113][114][115] ***just the electrical activity of a single neuron as computational-equation:***

1. **Ci * dVi/dt = -gL,i * (Vi - EL) + Isum** *(change in membrane potential of a neuron).*

2. **Isum = ∑ ji,u (Vi - Eu)gi,u + Iext** *(description of current flowing in the neuron).*

3. **dji,u / dt = -1/Tu * ji,u + ωu * ∑ t-tkm** *(open-channel percentage modified by pre-synaptic activity/decay.)*

Where:

**Ci** = Neuronal capacitance of neuron I

**dVi/dt** = Change in membrane potential of neuron i over time

**gL,i** = Leak conductance of neuron i

**Vi** = Membrane potential of neuron i

**EL** = Equilibrium (reversal) potential of leak channels

**Isum** = Sum of all synaptic and external currents into neuron

**ji,u** = Percentage of open channels for neurotransmitter u at synapse i

**(Vi - Eu)** = Driving force ofhe current, where Eu is equilibrium (reversal) potential for neurotransmitter u

**gi,u** = Conductance of synapse i for ions related to neurotransmitter u

**∑ ji,u (Vi - Eu)gi,u** = Sum of all synaptic currents

**u Є { AMPA, NMDA, GABAa, GABAb }** = Specific neurotransmitter type for the synapse i

**Iext** = External current applied to the neuron

**dji,u / dt** = Change in percentage of open channels for neurotransmitter u over time

**-1/Tu * ji,u** = Rate at which open channels close over time

**ωu** = Synaptic weight associated with neurotransmitter u

**ωu * ∑ t-tkm** = Influence of past presynaptic spikes (m) at time k, weighted by ωu

**t-tkm** = Time difference between current time t and past spike k of a pre-synaptic neuron m*

**m** = Set of indices of pre-synaptic neurons of neuron i

Pseudocode for ***just the electrical activity of a single neuron:***

(structured pipeline implemented in TensorFlow for tensor generation)

(actual *source*-code component_4 (DTB) is included in Appendix-2)

```
### DTB implementation (component_4) of 5 ###
##  "ai2agi by: ai70000, Ltd."   ##

## overview: create synthetic-data from DTB-equations, synthetic-data input into
TF-model, TF-model outputs tensors, tensors input into CV_adapt (component_5),
tensors for: components_1, _2, _3, and _4 are inputs to CV_adapt

DTB implementation (component_4)

## overview: pseudocode for synthetic-data and TF-model, DataGenerator class creates
synthetic neural data, (real data is used for true_AGI, DTBModel class implements a
TensorFlow-based neural model for step5, DTB neural-network-simulation creates:
digital twin of brain (DTB), DTB is 1-of-5 components for true_AGI

##pseudocode for synthetic-data and TF-model
## (main program implementation, 3 python-files: II, III, IV] follows below)

#synthetic data that incorporates all variables and equations:

# Components:

### Equations to Model

1. **Membrane Potential Change**: involves neuronal capacitance \( C_i \), leak
conductance \( g_{L,i} \), membrane potential \( V_i \).

2. **Synaptic and External Currents**: Summation of currents of various synapses \( j
\), each has their own neurotransmitter-specific conductance \( g_{i,u} \) and
equilibrium potential \( E_u \), and an external current \( I_{ext} \).

3. **Channel Dynamics**: Change in open-channel percentages \( j_{i,u} \) over time,
with decay governed by \( \tau_u \) and influenced by past presynaptic spike timings.
```

```
## Additional Variables
*Neuronal Capacitance** \( C_i \)
*Leak Conductance** \( g_{L,i} \)
*Equilibrium Potentials** \( E_L \) (for leak) and \( E_u \) (for each neurotransmitter)
*Synaptic Weight** \( \omega_u \)
*Presynaptic Spike Influence** \( t_{-tkm} \) (timing of spikes from presynaptic
neurons)
*Decay Rate** \( \tau_u \) for neurotransmitter channel dynamics
*Synaptic Current** \( I_{\text{syn}} \) for each neurotransmitter \( u \)


## Approach to Data Generation
#generate random data for all components.
# structure:

##1. **Neuronal Capacitance**
##2. **Presynaptic Spike Timing**
##3. **Channel Openings**
##4. **Synaptic Currents**

# script generates all synthetic data and returns a structured tensor suitable for
**cv_adapt** input.



# pseudocode for generating the synthetic data:

# Parameters for synthetic data generation
n_samples = 100  # Number of samples (time points)
n_neurons = 5    # Number of neurons (for example)
n_neurotransmitters = 4  # AMPA, NMDA, GABAa, GABAb
n_presynaptic_neurons = 3  # Number of presynaptic neurons affecting each neuron
```

```
# Random synthetic data for each of the neuron dynamics

# 1. Neuronal Capacitance (Ci)
C = np.random.uniform(1e-9, 1e-6, size=(n_samples, n_neurons, 1))  # Capacitance in
Farads

# 2. Leak Conductance (gL,i)
gL = np.random.uniform(1e-9, 1e-6, size=(n_samples, n_neurons, 1))  # Conductance in
Siemens

# 3. Membrane Potential (Vi)
V = np.random.uniform(-70, -50, size=(n_samples, n_neurons, 1))  # Membrane potential
in mV

# 4. Equilibrium Potential of Leak (EL)
EL = np.random.uniform(-80, -60, size=(n_samples, n_neurons, 1))  # Leak potential
in mV

# 5. Synaptic Currents (Isyn)
Isyn = np.random.uniform(-1e-9, 1e-8, size=(n_samples, n_neurons,
n_neurotransmitters))  # Current for each neurotransmitter

# 6. External Current (Iext)
Iext = np.random.uniform(-1e-9, 1e-8, size=(n_samples, n_neurons, 1))  # External
current

# 7. Channel Openings for each Neurotransmitter (ji,u)
ji_u = np.random.uniform(0, 1, size=(n_samples, n_neurons, n_neurotransmitters))  #
Channel opening %

# 8. Decay Time Constant for each Neurotransmitter (Tau_u)
Tau_u = np.random.uniform(1e-3, 1e-2, size=(n_samples, n_neurons,
n_neurotransmitters))  # Decay time in seconds

# 9. Synaptic Weights (ωu)
omega_u = np.random.uniform(1e-3, 1e-2, size=(n_samples, n_neurons,
n_neurotransmitters))  # Synaptic weight

# 10. Presynaptic Spike Influence (tkm)
tkm = np.random.uniform(0, 100, size=(n_samples, n_neurons,
n_presynaptic_neurons))  # Time of presynaptic spikes

# Stack all variables into a final input tensor for the DTB model
input_data = np.concatenate([C, gL, V, EL, Isyn, Iext, ji_u, Tau_u, omega_u, tkm],
axis=-1)

# Convert to TensorFlow tensor
input_tensor = tf.convert_to_tensor(input_data, dtype=tf.float32)
```

```
# Display first 5 samples of the tensor
print(input_tensor[:5])  # This will display the tensor structure
### Variables Explained

##- **C, gL, V, EL**: Each neuron will have individual capacitance, leak conductance, membrane potential, and equilibrium potential.

##- **Isyn, Iext**: Currents from multiple neurotransmitter types (AMPA, NMDA, GABAa, GABAb) and external currents.

##- **ji_u, Tau_u**: Dynamics for neurotransmitter channels (open percentage and decay).

##- **omega_u**: Synaptic weights for each neurotransmitter.

##- **tkm**: Time differences for presynaptic spikes, influencing the current state.

##The output is a tensor with shape:
(n_samples, n_neurons, total_features)` where `total_features` includes all the listed dynamic variables.

##This data now serves as input to the **CV_Adapt** module.


# Define DTB parameters as tensors
num_neurons = 100  # Example neuron count
num_synapses = 100  # Example synapse count

# Neuronal states
Vi = tf.Variable(tf.random.uniform([num_neurons], minval=-70.0, maxval=-50.0), dtype=tf.float32)  # Membrane potentials
ji_u = tf.Variable(tf.zeros([num_neurons, num_synapses]), dtype=tf.float32)  # Open channel percentages

# Constants
Ci = tf.constant(1.0, dtype=tf.float32)  # Neuronal capacitance
gL_i = tf.constant(0.1, dtype=tf.float32)  # Leak conductance
EL = tf.constant(-65.0, dtype=tf.float32)  # Equilibrium potential
Eu = tf.constant([-70.0, 0.0, -80.0, -90.0], dtype=tf.float32)  # Reversal potentials for AMPA, NMDA, GABAa, GABAb
gi_u = tf.Variable(tf.random.uniform([num_neurons, num_synapses]), dtype=tf.float32)  # Synaptic conductance
Iext = tf.Variable(tf.zeros([num_neurons]), dtype=tf.float32)  # External current
Tu = tf.constant(5.0, dtype=tf.float32)  # Decay constant
wu = tf.Variable(tf.random.uniform([num_synapses]), dtype=tf.float32)  # Synaptic weight
```

```
# Compute Isum (sum of synaptic and external currents)
Vi_expanded = tf.expand_dims(Vi, axis=1)  # Expand for broadcasting
drive_force = Vi_expanded - Eu  # Driving force
synaptic_currents = tf.reduce_sum(ji_u * drive_force * gi_u, axis=1)  #

Compute sum of all synaptic currents
Isum = synaptic_currents + Iext

# Compute membrane potential update
dVi_dt = (-gL_i * (Vi - EL) + Isum) / Ci
Vi_new = Vi + dt * dVi_dt

# Compute neurotransmitter channel dynamics
dji_u_dt = (-1.0 / Tu) * ji_u + wu * tf.reduce_sum(tf.exp(-dt))  # Placeholder for
spike influence
ji_u_new = ji_u + dt * dji_u_dt

# Tensor update operation
update_op = [Vi.assign(Vi_new), ji_u.assign(ji_u_new)]

# Function to run DTB step
def dtb_step():
    tf.function(lambda: tf.group(*update_op))()

# Example execution for 10 steps
for _ in range(10):
    dtb_step()

print("DTB step execution complete.")
```

***just the electrical activity of a single neuron*** summary:

- **C, gL, V, EL**: Each neuron will have individual capacitance, leak conductance, membrane potential, and equilibrium potential.
- **Isyn, Iext**: Currents from multiple neurotransmitter types (AMPA, NMDA, GABAa, GABAb) and external currents.
- **ji_u, Tau_u**: Dynamics for neurotransmitter channels (open percentage and decay).
- **omega_u**: Synaptic weights for each neurotransmitter.
- **tkm**: Time differences for presynaptic spikes, influencing the current state.
- output is a tensor with shape `(n_samples, n_neurons, total_features)` where `total_features` includes all the listed dynamic variables.

**Explanation of *just the electrical activity of a single neuron* pseudocode**

This pseudocode outlines DTB component, its primary function is to create a digital twin brain (DTB) through neural-net simulation. The approach generates synthetic data based on established neuronal equations and then feeds this data into a TensorFlow model. DTB output is then used as input for CV_Adapt.

DTB initialized by creating synthetic data based on fundamental neuronal equations. The equations model key aspects of neuronal dynamics. This includes membrane potential changes, synaptic currents, and channel dynamics. The variables involved in these equations are randomly generated. This simulates the variability and complexity of real neural data.

The generated data is structured as a tensor. The tensors have dimensions representing samples, neurons, and features. The structured tensor is converted to TensorFlow tensors.

DTB model implemented using TensorFlow variables and constants. These represent parameters of the model; membrane potentials, channel openings, and synaptic conductances. The code calculates the sum of synaptic and external currents. This step models the neuronal dynamics. The membrane potential is updated based on the calculated currents and other parameters (see appendix.) The channel openings are updated. This models the dynamics of neurotransmitter channels.

***Now add: chemical, glial-cell, brain-region, connectome, dynamic-activity, and internal focus elements to just the electrical activity of a single neuron, and scale from a single neuron to the fully complete set of billions of neurons*** (Gerstner et al. (2014) [50], Dayan & Abbott, 2001 [49], Markram et al., (1998))[121]. Scaling up from just the electrical activity of a single neuron to the entire brain (represented initially as billions of neurons with trillions of synapses with other components added later) involves a massive increase in complexity (Sporns, (2011) [68], Koch & Segev (1998) [122], Izhikevich et al., (2004) [123].

There's no single equation which captures the electrical activity of the whole brain. Challenges include a) scaling up to billions of neurons, each with thousands of connections (computationally demanding to model), b) adding many other forces to the neuron other than just the electrical signals deciphered in the previous section (interactions between neurons, neurotransmitters, glial cells, and regions), and c) understanding the precise wiring diagram (the connectome) [68], (Marder, (2012) [94], (Allen & Eroglu, (2017))[124].

Approaches to brain modeling include i) Network Models of interconnected nodes (neurons or brain regions), with connections between nodes defined by connection matrices, representing strength of interactions, and simulation of the spread of activity across the network [68]. ii) Mean-Field Models which simplify complexity by averaging activity of large populations of neurons via differential equations to describe average firing rate or membrane potential of a population (Wilson & Cowan, 1972) [126]. iii) Computational Neuroscience and Simulation to simulate detailed models of brain circuits and incorporate realistic neuron models, synaptic plasticity, and network connectivity (with software: NEURON and Brian) (Carnevale & Hines, 2006 [89], Goodman & Brette, 2009 [97]. iv) Dynamic Causal Modelling to infer connectivity between brain regions from neuroimaging fMRI and EEG data, with math models to describe activity of one region influencing others (Friston et al., 2003) [81].

And all this is done with; Connectivity Matrices which represent the strength of connections between neurons or regions (Sporns, 2011)[68], Oscillations at different frequencies (e.g., alpha, beta, gamma) (Buzsáki & Draguhn, 2004) [127], and Spatiotemporal Dynamics to model patterns of changes over space and time (Breakspear, 2017) [128]. So modeling the brain requires a combination of math models, computational simulations, and neuroimaging data. But we should do this because if we want to achieve AGI within the ~7 years, we need to include the DTB component. This in part is why there's a global arms race in AI and AGI and why there is going to be an explosion in the construction of very large AI data-centers and AI-data-center warehouses.

With that introduction for how we go from just the electrical activity of our single neuron, to adding: chemical, glial-cell, brain-region, connectome, dynamic-activity and internal focus to our single neuron, and to then scaling our single neuron to billions of neurons, we proceed.

Neurotransmitter ***chemical*** [48][49][50] messengers mediate communication between neurons:

Equation (Placeholder): [Neurotransmitter Release] = f([Pre-synaptic Activity], [Neurotransmitter Availability]) where, function 'f' = relationship between pre-synaptic neuron activity, neurotransmitter availability, and amount of neurotransmitter released.

Equation (placeholder): [Neurotransmitter Effect on Postsynaptic Receptor] = g([Neurotransmitter Concentration], [Receptor Sensitivity]) where, function 'g' models effect of neurotransmitter concentration on postsynaptic receptors, influencing synaptic conductance.

***Glial Cells*** astrocytes, microglia, and oligodendrocytes play a role in brain function. [83].

Equation (Placeholder): [Glial Influence on Synaptic Activity] = h([Glial Activity], [Synaptic Activity]) where function 'h 'represents how Glial cells modulate synaptic transmission.

Equation (placeholder): [Glial influence on ion concentration] = j([ion concentration],[Glial activity]) where function 'j' represents how Glial cells modulate the ion concentration within the extracellular space.

***Brain Regions*** Different regions have specialized functions and interact through anatomical connections. [84] We represent these interactions using connection matrices in network models.

Equation (Placeholder): [Regional Activity] = Network Model ([Neuron Activities], [Connection Strengths]) which represents combined activity of neurons within a specific brain region.

***Connectivity (Connectome)*** is the brain's wiring diagram. Recent advances use diffusion MRI and tractography to map brain connections. [51][52] Modern connectome diagrams visualize the brain as a network of nodes (brain regions) connected by edges (fiber tract)

***Dynamic Activity*** and External/Internal Factors

External Sensory input (visual, auditory, etc.)

Equation (Placeholder): [Sensory Input Effect] = k([Sensory Stimulus], [Neural Sensitivity]) where function 'k' represents how sensory stimuli influence neural activity.

***Internal Attention,*** emotions, sleep-wake cycles, etc.

Equation (Placeholder): [Internal State Effect] = l([Internal State Variable], [Neural Modulation])

function 'l' represents how internal states modulate neural activity.

Equation (placeholder): [Neuromodulator effect] = m([neuromodulator concentration], [neuronal receptor sensitivity]) function 'm' represents how neuromodulators, such as dopamine or serotonin, affect neuronal activity.

Incorporating these elements into our ***just the electrical activity of a single neuron***

Incorporating these elements into our just the electrical activity of a single neuron Main Equation, we add these factors as additional terms to the single-neuron equation and or use them to modulate parameters.

This builds upon the foundational work of Hodgkin and Huxley (1952) [48], who first quantitatively described membrane current and its application to conduction and excitation in nerve. So now we added all the other 'stuff' to our single neuron, and our refactored Single-Neuron Equation (with Placeholders) is:

**$C_i * \frac{dV_i}{dt} = -g_{L,i} * (V_i - E_L) + \sum j_{i,u} (V_i - E_u) g_{i,u} + I_{ext}$**

**+ k([Sensory Stimulus], [Neural Sensitivity])**

**+ l([Internal State Variable], [Neural Modulation])**

**+ m([neuromodulator concentration], [neuronal receptor sensitivity])**

**+ h([Glial Activity], [Synaptic Activity])**

And now we have our Network Model Implementation For a network of N neurons, where we have N such equations, one for each neuron. This type of neural network modeling is extensively discussed in works such as Dayan and Abbott (2001)[49] and Gerstner et al. (2014) [50]. The synaptic currents (ji,u) depend on the activity of pre-synaptic neurons. The dynamics of these synaptic connections, including variability and strength, are explored in research by (Markram et al. (1998)[121] and Tsodyks et al. (1998) [129] . Connection matrices define the strength of these synaptic connections. The output of these equations is fed into: the equations for the Glial cells, the neurotransmitter equations, and the regional activity equations. The importance of glial cells in synaptic activity, forming tripartite synapses, is highlighted by Araque et al. (1999) [313], and the cellular mechanisms of astrocyte-synapse interaction is reviewed by Allen and Eroglu (2017) [124].

(These placeholder equations are conceptual and need to be filled in with specific mathematical functions and parameters however for the purposes of our foundational blueprint for our DTB, and with our DTB, growing it from a single neuron with just electrical activity to a fully functional complete neuron, and then scaling that to the full complement of a billions of neurons, and then scaling those billions of neurons to trillions of synapses, and then adding all the extra functionality that goes along with all that, we're ok from the perspective of computer-science to build out our preliminary infrastructure with those placeholders, at least until the neuroscience team says otherwise.)

The terms k, l, m, and h are placeholders. They represent influences on the neuron, but don't fully define the complex dynamics of those influences. To fully model those elements, separate, detailed equations for neurotransmitter release, and glial cell activity are required, and then that is fed into this primary equation. The concept of neuromodulation, represented by term 'm', is discussed in reviews by Nadim and Bucher (2014) [107] and Marder (2012) [94]. But we're ok to move forward with the understanding that those placeholders will be filled in later.

**Scaling from a single neuron to billions of neurons (for the DTB)**

Scaling the single-neuron model to represent the entire brain involves moving from our single equation to a network of interconnected equations [59][60][61].

To do this you use neuron-indexing. An index 'i' represents each of the N (billions) neurons in the brain. Each neuron 'i' has its own set of variables: Vi (membrane potential), and ji,u (synaptic conductances). [51, 52]

To do this you also use synaptic-connections: A connection matrix 'Wij' [48, 49, 50] represents the strength of the synaptic connection from neuron 'j' to neuron 'i'. This matrix is very large (N x N) and likely very sparse (most connections are zero). In the matrix not all connections are equal. Some are excitory, others are inhibitory.

And you use synapse-indexing because there's trillions of synapses, and *each* synapse requires indexing.

This is done by indexing the pre-synaptic and postsynaptic neuron.

So this gets us our Membrane Potential Equation (for each neuron i):

**Ci * dVi/dt = -gL,i * (Vi - EL)**

**+ ∑ (∑ jji,u (Vi - Eu)gi,u) + Iext,i**

**+ ki([Sensory Stimulus], [Neural Sensitivity])**

**+ li([Internal State Variable], [Neural Modulation])**

**+ mi([neuromodulator concentration], [neuronal receptor sensitivity])**

**+ hi([Glial Activity], [Synaptic Activity])**

Where:

'i' now indexes each individual neuron.

The *double summation:* **+ ∑ (∑ jji,u (Vi - Eu)gi,u) + Iext,i**

now shows that all synapses, and all synapse types are being accounted for.

Iext,i, ki, li, mi, and hi are now specific to each neuron.

And this gets us our Synaptic Conductance Equation [51, 52] (for each synapse) [48, 49, 50]:

**$dj_{i,u} / dt = -1/T_u * j_{i,u} + \omega_u * \sum (W_{ij} * spikes_j(t-t_{km}))$**

Where:

'j' indexes presynaptic neurons.

'Wij' is the connection strength from neuron 'j' to neuron 'i'.

spikesj(t-tkm) represents the timing of pre-synaptic spikes from neuron j.

Neurotransmitter Equations (for each synapse) These equations are implemented for each synapse, and each neurotransmitter type. These equations calculate the amount of neurotransmitter released, and the effect of that neurotransmitter (Marder, 2012) [94].

Glial Cell Equations (for each glial cell): Glial cells are indexed. These equations calculate the activity of each glial cell, and the impact that each glial cell has on the surrounding neurons (Araque et al., 1999; Allen & Eroglu, 2017) [130][125].

Regional Activity Equations calculate the average activity of groups of neurons, that represent brain regions. These equations take the output of the individual neuron equations as an input (Dayan & Abbott, 2001; Gerstner et al., 2014) [49][50].

Internal and External Factor Equations use outside data, and internal state data, to modulate the other equations (Nadim & Bucher, 2014) [107].

Computational Challenges: Massive Matrices The connection matrix 'Wij' would be enormous, requiring vast memory. Sparse matrix techniques would be essential to reduce storage requirements. Computational Power Solving coupled differential equations for billions of neurons and trillions of synapses would require super-computing. Parallel processing and efficient algorithms would be crucial. Data Requirements Accurate connectome data, neurotransmitter dynamics, and glial cell interactions are needed, obtaining this data is a challenge. Mean-Field Approximations Instead of simulating individual neurons, you model the average activity of neuronal populations to reduce computational load (Dayan & Abbott, 2001; Gerstner et al., 2014) [49][50].

Reduced Models Simplified neuron models (e.g., leaky integrate-and-fire) used to reduce complexity.

Hierarchical Modeling brain modeled at multiple levels of abstraction (e.g., neurons, microcircuits, brain regions).

***These equations, specifically the indexing of 'i' for each neuron and the Wij*** [48, 49, 50] ***connection matrix, are the core components that scale the single-neuron building block to billions of neurons.***

**Ci * dVi/dt = -gL,i * (Vi - EL)**

**+ ∑ (∑ jji,u (Vi - Eu)gi,u)**

**+ Iext,i + ki([Sensory Stimulus], [Neural Sensitivity])**

**+ li([Internal State Variable], [Neural Modulation])**

**+ mi([neuromodulator concentration], [neuronal receptor sensitivity])**

**+ hi([Glial Activity], [Synaptic Activity])**

**dji,u / dt = -1/Tu * ji,u + ωu * ∑ (Wij * spikesj(t-tkm))**

***The part of the equations that scales to trillions of synapses is primarily within the synaptic current term and the synaptic conductance equation:***

**∑ (∑ jji,u (Vi - Eu)gi,u):**

The double summation (∑) is crucial. The inner ∑ sums across all synapse types (u), and the outer ∑ effectively sums across all synapses connected to neuron 'i', which will be in the thousands. ***When this is done for all billions of neurons, this results in trillions of synaptic calculations.***

**dji,u / dt = -1/Tu * ji,u + ωu * ∑ (Wij * spikesj(t-tkm)):**

The indexing of 'ji,u' means that there is a conductance value for each synapse.

The term Wij * spikesj(t-tkm) represents the effect of each pre-synaptic spike on the postsynaptic conductance. The Wij connection matrix, and the summing of pre-synaptic spikes, means that each synapse connection is calculated. This dynamic synaptic activity is essential for information processing in neural circuits (Markram et al., 1998) [121].

A true "overview" macro equation for a Digital Twin of the Brain (DTB) should conceptually represent the interaction with the external world and the body. This includes: Sensory Input: how the DTB receives information from the senses (Gerstner et al., 2014) [50], Motor Output: how the DTB generates actions that affect the body and the environment (Dayan & Abbott, 2001) [49], Hormonal/Chemical Signaling from the Body: the brain is highly affected by the body's current state (Nadim & Bucher, 2014). [107].

**Quantification of Sensory Input, Motor Output, Hormonal/Chemical Signals as factors integrated into our cumulative DTB equation**

**1. Sensory Input (S(t))**

S(t) is a multi-dimensional array representing processed sensory data at time 't'. Dimensions include: Time, Sensory Modalities, and Feature Space. For instance, visual input might be a tensor representing (time, pixel coordinates, RGB values) or (time, identified objects, object properties), while audio input might be (time, frequency bands, amplitude). This concept of processing sensory data into meaningful features aligns with research on neural coding and sensory processing (Dayan & Abbott, 2001) [49].

Connection to DTB: S(t) acts as an external input to the DTB's neural network, modifying the Iext,i (external current) term in the neuron equations and modulating sensory sensitivity in the ki term. The DTB's neural activity patterns are shaped by S(t), influencing perception and cognition. This process of external input driving neural activity is fundamental to how neural networks process information (Gerstner et al., 2014)[50]. The Tensor data from the CV_ADAPT portion of the AGI, acting as AGI-DNA replication, will provide the processed sensory data for S(t).

**2. Motor Output (M(t))**

M(t) is a multi-dimensional array representing motor commands at time 't'. Dimensions include: Time, Motor Effectors, and Action Parameters. For example, robotic arm control might be a tensor (time, joint angles, motor torques), and speech output might be (time, phoneme sequence, intonation).

Connection to DTB: M(t) is generated by the DTB's neural network. Specific neural populations within the DTB encode motor commands, which are translated into M(t) through motor control models. This process of neural signals driving motor actions is a core area of study in motor control and computational neuroscience (e.g., see theoretical approaches in Dayan & Abbott, 2001). The DTB's motor output will drive the CV_ADAPT portion of the AGI, enabling it to interact with its environment.

**3. Hormonal/Chemical Signaling (H(t))**

H(t) is a vector representing the concentrations of key hormones and chemicals at time 't'. Elements include: Concentrations of hormones (e.g., cortisol, dopamine, serotonin), levels of neurotransmitter precursors or metabolites, and other relevant chemical signals.

Connection to DTB: H(t) modulates the DTB's neural activity and synaptic plasticity. It affects the mi (neuromodulator effect) term in the neuron equations. Hormones can alter receptor sensitivity, synaptic strength, and neuronal excitability, as discussed in research on neuromodulation (Nadim & Bucher, 2014; Marder, 2012)[107] [94]. H(t) provides a feedback loop, where the body's state influences the brain's activity and vice versa, highlighting the complex interaction between physiological states and neural function. The data for the H(t) vector will be gathered from the body and processed for use by the DTB.

**Quantification of Sensory Input, Motor Output, Hormonal/Chemical Signals**

(sensory input, motor output, and hormonal/chemical signaling) is generated and fed to DTB.

To generate the sensory input (S(t)) for our AGI, we need to capture data from its environment using real-world sensors: cameras, microphones, and tactile sensors.

Initially AGI exists in a simulated environment, so the simulation engine provides sensory data. Alternatively, pre-recorded datasets for initial development will work. This raw sensory data is converted into tensors by CV_ADAPT, via preprocessing, feature extraction, and converting data into the appropriate format and or data types (primarily tensors.)

For motor output (M(t)), the neural-net in DTB generates patterns of activity representing motor commands. A motor control model translates the patterns into specific actions, defining neural activity maps as muscle groups or actuator movements and calculating necessary parameters like force and velocity. These commands and parameters are then organized into the M(t) tensor, with the format depending on AGI's specific motor effector's.

To incorporate hormonal and chemical signaling (H(t)), simulation of the production, release, and transport of hormones and chemicals (once AGI has a simulated body,) is implemented.

Once it it interacts with a physical body, it can use biofeedback sensors to measure physiological signals: heart rate variability, skin conductance. etc...External data from wearable sensors and medical devices provides relevant information. This raw data is processed to extract hormone and chemical concentrations, and then organized into the H(t) vector.

When generating this data, data synchronization is required to maintain high data quality through accurate sensors. This will be as close to realism in simulated environments or bodies as it will achieve, to facilitate as AGI exists while learning and adapting to the real-world.

***These tensors are the interfaces between DTB and the external world/body. They represent complex, dynamic information processed by DTB's neural-net. The DTB's internal state (neural activity) is shaped by S(t) and H(t), and generates M(t) to interact with the environment. CV_ADAPT of AGI allows conversion of the data, to the correct tensor format, and is what takes DTB's output and converts it to actions***.

So now we've built out our single neuron from just the electrical activity to all functions [48, 49, 50], and then scaled it to billions of neurons [53, 54] and then scaled all that to trillions of synapses [51, 52], and we integrated Quantification of Sensory Input, Motor Output, Hormonal/Chemical Signals [33, 34, 35, 38].

Now let's break down how raw data for Quantification of Sensory Input, Motor Output, Hormonal/Chemical Signals (sensory input, motor output, and hormonal/chemical signaling) is generated and fed to DTB. To generate the sensory input (S(t)) for our AGI, we need to capture data from its environment using real-world sensors: cameras, microphones, and tactile sensors [32, 43].

Initially AGI exists in a simulated environment, so the simulation engine provides sensory data [33, 34, 35, 38]. Alternatively, pre-recorded datasets for initial development will work. This raw sensory data is converted into tensors by CV_ADAPT, via preprocessing, feature extraction, and converting data into the appropriate format and or data types (primarily tensors) [27, 28, 29, 30, 31].

For motor output (M(t)), the neural-net in DTB generates patterns of activity representing motor commands [33, 34, 35, 38]. A motor control model translates the patterns into specific actions, defining neural activity maps as muscle groups or actuator movements and calculating necessary parameters like force and velocity. These commands and parameters are then organized into the M(t) tensor, with the format depending on AGI's specific motor effector's.

To incorporate hormonal and chemical signaling (H(t)), simulation of the production, release, and transport of hormones and chemicals (once AGI has a simulated body) is implemented [57, 58]. Once it interacts with a physical body, it can use biofeedback sensors to measure physiological signals: heart rate variability, skin conductance, etc. External data from wearable sensors and medical devices provides relevant information. This raw data is processed to extract hormone and chemical concentrations, and then organized into the H(t) vector.

When generating this data, data synchronization is required to maintain high data quality through accurate sensors. This will be as close to realism in simulated environments or bodies as it will achieve, to facilitate as AGI exists while learning and adapting to the real-world.

It's exciting how quickly the DTB component has progressed from just one neuron of electrical activity [48][49][50] and at this stage if you're still with me I trust and hope you understand why the first three components were not as detailed as DTB, and CV_Adapt in the next section because on the trajectory from ai_LLMs to AGI, a possible new frontier is DTB and CV_Adapt, as outlined in this project paper as a possible way forward [1][3][5][7][9][14][23][24][37][43][44].

As the AGI adapts to the real world a major part of its initial learning will be to simply watch television [16, 17, 18, 19]. Television will be a valuable tool for AGI for learning if used thoughtfully and strategically so AGI will be able to generalize, make inferences, and understand the world.

To summarize the most recent integration to DTB: sensory input to DTB is modeled by the equation: $I_{sensory, i}(t) = k(S(t), Si)$, where $I_{sensory, i}(t)$ represents the sensory input current to neuron 'i' at time 't'].

The function 'k' maps the sensory input tensor S(t) to a current value, taking into account the neuron's sensory modality, extracting relevant features from sensory data, and applying a receptive field. S(t) represents the sensory input tensor, and Si a set of parameters specific to neuron 'i' that determine its sensitivity to sensory features.

Motor output from DTB is represented by the equation $M(t) = G(A(t))$, where M(t) is the motor output tensor at time 't'. The function 'G' maps neural activity patterns to motor commands, potentially using population coding, motor primitives, or inverse kinematics. A(t) represents the activity of motor-related neurons in DTB.

Hormonal / chemical signaling is modeled by the equation H'(t) = f_h(H(t), N(t)), where H'(t) is the rate of change of the hormonal/chemical state vector at time 't' [57][ 58].

The function f_h simulates the dynamics of the hormonal/chemical system, including production/degradation, diffusion/transport, and interactions between different hormones and chemicals. H(t) represents the hormonal/chemical state vector, and N(t) represents the activity of neurons involved in hormonal/chemical signaling.

These equations are integrated with DTB by incorporating sensory input current into the neuron's membrane potential equation, using the motor output equation to generate actions, and updating the hormonal/chemical state vector based on the hormonal/chemical signaling equation, which modulates DTB's neural activity. This formalized representation captures key processes of sensory processing, motor control, and hormonal/chemical modulation in DTB's interaction with the external world or body.

**Computational implementation of DTB with a symbolic equation:**

DTB_simulation = F ( {N}, {S}, {C}, {P}, H, E, A )

Where:

- DTB_simulation: Represents running simulation of Digital Twin Brain.
- F: simulation framework encapsulates software environment (e.g., NEURON, Brian, custom code), numerical methods used to solve differential equations (Euler, Runge-Kutta), and hardware infrastructure (CPU's, GPUs, memory). It's the 'engine' that drives DTB, just like CV_Adapt is the 'processor' of AGI.
- {N}: set of all neuron objects, each with state variables (membrane potential, synaptic conductances, etc.).
- {S}: plasticity parameters.
- {C}: set of all connection objects, representing network topology and connection strengths between neurons and regions.
- {P}: set of all parameters governing DTB, including neuron properties, synaptic properties, plasticity rules, regional parameters, and simulation settings.
- H: hardware resources allocated to simulation, including processing power, memory, and storage.
- E: external environment or simulation environment DTB interacts with.
- A: algorithms used for data input/output, sensory processing, motor control, and other computational tasks.

The computational implementation of the Digital Twin Brain (DTB) can be represented by the symbolic equation: DTB_simulation = F ( {N}, {S}, {C}, {P}, H, E, A ). This equation encapsulates the complex interplay between biological components and computational infrastructure. The 'F' term, representing the simulation framework, encompasses software environments like NEURON (Carnevale & Hines, 2006) [59] and Brian (Goodman & Brette, 2009)[60], as well as the high-performance computing resources necessary for large-scale simulations (Izhikevich, Edelman, & Krichmar, 2004) [123]. The sets {N}, {S}, {C}, and {P} represent the biological components and their parameters. Hardware resources, 'H', are crucial for efficient simulation, while 'E' and 'A' represent the interaction with the external environment and the algorithms used for data processing.

**Equation to capture the essence of External World Interaction:**

$E_{interaction}(t) = I( S(t), M(t), B, W(t) )$

Where:

- $E_{interaction}(t)$: overall interaction between DTB and external world at time 't'. This encompasses both incoming sensory information and outgoing motor actions.
- I: symbolizes interaction function. It defines how DTB processes sensory input, generates motor output, updates internal representation of the world. It's the "interface" between DTB and the environment.
- S(t): sensory input tensor at time 't', as previously defined. This is the raw data DTB receives from the world.
- M(t): motor output tensor at time 't', as previously defined. This represents the actions DTB takes to influence the world.
- B: embodiment of DTB. A physical robot, simulated agent in a virtual environment, or abstract entity with defined ways of sensing and acting. The key: it provides a means for DTB to interact with the world.
- W(t): DTB's world model at time 't'. This is DTB's internal representation of the external world, including its knowledge of objects, properties, relationships, and dynamics. It's updated based on sensory input and the consequences of the DTB's actions.

Placeholder / algebraic variables:

- Neuron Variability
- Subcellular Processes
- Neurotransmitter Complexity
- Consciousness and Subjectivity
- Higher-Level Cognition

To capture intricate details of individual neuron variability, variables $\tau_i$ (membrane time constant), $\theta_i$ (spike threshold), $\rho_i$ (refractory period), and $M_i$ (morphology parameters) are used.

These variables integrate into the model by incorporating $\tau_i$ into the membrane potential equation, $\theta_i$ in a spiking mechanism, $\rho_i$ preventing immediate firing after a spike, and $M_i$ more complex compartmental models of the neuron (Destexhe, Rudolph, & Paré, 2003) [131].

Sub-cellular processes are represented by $G_{i,p}(t)$ (gene expression level), $P_{i,q}(t)$ (protein concentration), and $S_{i,r}(t)$ (state of signaling pathway). These variables integrate by linking gene expression to ion channel production, connecting protein levels to synaptic plasticity mechanisms, signaling pathways to modulate neuronal excitability or neurotransmitter release. This integration adds differential equations to model the dynamics of these sub-cellular processes. Neurotransmitter complexity is variables $N_{i,u,v}(t)$ (neurotransmitter concentration), $R_{i,u,w}(t)$ (receptor state), and $T_{i,u}(t)$ (transporter activity). These variables integrate by expanding synaptic conductance to include multiple neurotransmitters and their interactions, modeling receptor dynamics to account for binding, unbinding, and desensitization, and incorporating transporter activity to influence neurotransmitter clearance from the synapse (Marder, 2012) [94].

Consciousness and subjectivity are C_i(t) (consciousness index) and Φ(t) (global consciousness field). Integration involves C_i by factors complexity of neural activity and feedback connections, Φ(t) emerges from collective network activity and influences individual neuron behavior (Tononi, 2008) [95].

Higher-level cognition is represented with variables L_i(t) (linguistic representation), P_i(t) (planning variables), and D_i(t) (decision-making variables). These are integrated by incorporating L_i into language processing, using P_i in prefrontal cortex regions to model planning, and D_i in decision-making circuits to simulate choices and actions (Miller & Cohen, 2001) [96].

Implementing a complex Digital Twin Brain (DTB) model requires a robust computational framework (Brette et al., 2007) [97]. A simulation environment, such as NEURON, Brian, or NEST, and a custom framework written in Python or C++. The appropriate data structures to represent neurons with their state variables, parameters, and connectivity, synapses with their connection information and plasticity rules, brain regions with their constituent neurons and connectivity patterns, and the overall network structure using graphs or connectivity matrices need source code by software-engineers and ai-engineers, code assistance to do some of the early heavy lifting is ok but to get all the teams and their projects to integrate, and to make sure the individual projects are contained, and the most importantly, DTB, CV_Adapt and AGI are contained, it will require substantial oversight layers."

The equations developed provide a foundation for DTB. Bridging the gap requires ongoing research and development in neuroscience, AI, and computer science. But there is a final element to DTB which needs to be added: a list of algebraic variables which are required for more advanced features once the project reaches that point. By including these placeholders now it will allow for better efficiency for scaling of the project without having to go all the way back and re-engineer these features from scratch.

Developing the algorithms for numerical integration to solve the differential equations governing the DTB's dynamics, simulate synaptic transmission, implement plasticity rules to update synaptic weights, compute regional activity, and handle efficient data input/output is also required. For large-scale simulations, high-performance computing resources, GPU acceleration, DLA's, and memory-efficient data structures and algorithms need to be integrated.

This foundational blueprint for DTB, as component_4 of AGI, represents one of five components comprising a synergistic foundational blueprint as one of many possible incremental paths forward to achieving true AGI by 2032. [2] [3] [5][7][9][14][15][23][24] [27][30][31][37][43][44][82][102][112][113][114].

**(DTB) component.**

**Key Concepts and Components:**

- **DTB:** Digital Twin Brain
- **Neuronal Activity:** Membrane potential, synaptic currents, etc.
- **Neurotransmitters:** Chemical messengers.
- **Glial Cells:** Support cells.
- **Connectome:** Brain's wiring diagram.
- **Sensory Input (S(t)):** External data.
- **Motor Output (M(t)):** Actions.
- **Hormonal/Chemical Signaling (H(t)):** Internal state.
- **Network Models:** Interconnected neurons.
- **Mean-Field Models:** Average neuronal activity.
- **Computational Neuroscience:** Simulation.
- **Dynamic Causal Modelling:** Inferring connectivity.

Mathematical Notation:

- ✓ Single Neuron Electrical Activity:
  - **$Ci * dVi/dt = -gL,i * (Vi - EL) + Isum$**
  - **$Isum = \sum ji,u (Vi - Eu)gi,u + Iext$**
  - **$dji,u / dt = -1/Tu * ji,u + \omega u * \sum (Wij * spikesj(t-tkm))$**
- ✓ DTB as a Function:
  - **$DTB\_simulation = F ( \{N\}, \{S\}, \{C\}, \{P\}, H, E, A )$**
- ✓ External World Interaction:
  - **$E\_interaction(t) = I( S(t), M(t), B, W(t) )$**
- ✓ Sensory Input:
  - **$I\_sensory, i(t) = k(S(t), Si)$**
- ✓ Motor Output:
  - **$M(t) = G(A(t))$**
- ✓ Hormonal/Chemical Signaling:
  - **$H'(t) = f\_h(H(t), N(t))$**
- ✓ Scaled Neuron Equations:
  - **$Ci * dVi/dt = -gL,i * (Vi - EL) + \sum (\sum jji,u (Vi - Eu)gi,u) + Iext,i + ki([Sensory Stimulus], [Neural Sensitivity]) + li([Internal State Variable], [Neural Modulation]) + mi([neuromodulator concentration], [neuronal receptor sensitivity]) + hi([Glial Activity], [Synaptic Activity])$**
  - **$dji,u / dt = -1/Tu * ji,u + \omega u * \sum (Wij * spikesj(t-tkm))$**
- ✓ **Neurotransmitter Release:**
  - **[Neurotransmitter Release] = f([Pre-synaptic Activity], [Neurotransmitter Availability])**

- ✓ Neurotransmitter Effect:
  - [Neurotransmitter Effect on Postsynaptic Receptor] = g([Neurotransmitter Concentration], [Receptor Sensitivity])
- ✓ Glial influence on synaptic activity:
  - [Glial Influence on Synaptic Activity] = h([Glial Activity], [Synaptic Activity])
- ✓ Glial influence on ion concentration:
  - [Glial influence on ion concentration] = j([ion concentration],[Glial activity])
- ✓ Regional activity:
  - [Regional Activity] = Network Model ([Neuron Activities], [Connection Strengths])
- ✓ Sensory Input Effect:
  - [Sensory Input Effect] = k([Sensory Stimulus], [Neural Sensitivity])
- ✓ Internal State Effect:
  - [Internal State Effect] = l([Internal State Variable], [Neural Modulation])
- ✓ Neuromodulator effect:
  - [Neuromodulator effect] = m([neuromodulator concentration], [neuronal receptor sensitivity])

**Mathematical Notation**

- ✓ Discretization:

Replace continuous derivatives (dVi/dt, dji,u/dt) with discrete approximations:

1. **dVi/dt ≈ (Vi_new - Vi) / dt**
2. **ji,u/dt ≈ (ji_u_new - ji_u) / dt**

- ✓ Code Simplification:
  - Code simplifies the spike influence term **(∑ (Wij * spikesj(t-tkm))) to wu * exp(-dt)** for computational efficiency. This is a simplification of the spike influence.
  - The j index is implicit in the ji_u matrix, and Wij is not represented explictly, rather it's represented as the random weights wu.
- ✓ Substitute discrete derivatives into original equations:

1. **Ci * (Vi_new - Vi) / dt = -gL,i * (Vi - EL) + Isum**
2. **(ji_u_new - ji_u) / dt = -1/Tu * ji_u + wu * exp(-dt)**

- ✓ Rearrange equations to solve for Vi_new and ji_u_new:

1. **Vi_new = Vi + dt * (-gL,i * (Vi - EL) + Isum) / Ci**
2. **ji_u_new = ji_u + dt * (-1/Tu * ji_u + wu * exp(-dt))**

✓ Map variables:

1. **gL,i → gL_i**
2. **ωu → wu**
3. **∑ (Wij * spikesj(t-tkm)) → wu * exp(-dt)**
4. **ji,u → ji_u**

Transitioning from continuous differential equations to discrete time-step approximations. The code simplifies complex biological functions (spike influence) for computational efficiency, which is represented in the simplified wu * exp(-dt) The code then solves for the next time step values. The notation shows how the core DTB equations are implemented in the code through discretization, simplification, and variable mapping

**Mathematical Notation:**

✓ Data Generator (DG):

- **DG(n_samples, n_neurons, n_neurotransmitters, n_presynaptic_neurons) → Tensor(C, gL, V, EL, Isyn, Iext, ji_u, Tau_u, omega_u, tkm)**
- Where: **C, gL, V, EL, Isyn, Iext, ji_u, Tau_u, omega_u, tkm** are matrices of randomly generated values within specified ranges.

✓ DTB Model (DM) Initialization:

- DM(num_neurons, num_synapses, **dt) → {Vi, ji_u, Ci, gL_i, EL, Eu, gi_u, Iext, Tu, wu}**
- Where: Vi, ji_u, gi_u, wu are initialized as TensorFlow Variables, and, Ci, gL_i, EL, Eu, Tu are initialized as TensorFlow Constants.

✓ **DTB Step (DS):**

- **DS({Vi, ji_u, Ci, gL_i, EL, Eu, gi_u, Iext, Tu, wu}, dt) → {Vi_new, ji_u_new, Isum}**
- Where:

**Isum = ∑(ji_u * (Vi - Eu) * gi_u) + Iext**

**dVi/dt = (-gL_i * (Vi - EL) + Isum) / Ci**

**Vi_new = Vi + dt * dVi/dt**

**dji_u/dt = (-1/Tu) * ji_u + wu * exp(-dt)**

**ji_u_new = ji_u + dt * dji_u/dt**

✓ Complete DTB Simulation:

- DTB_Simulation = Iterate(**DS, n_steps, {Vi, ji_u, Ci, gL_i, EL, Eu, gi_u, Iext, Tu, wu}, dt**)
- Where: Iterate is a function that performs n_steps iterations of the DS function, updating the state variables {Vi, ji_u} in each step.

Where:

✧ **DataGenerator (DG):** Generates synthetic neural data // `generate_data()`
✧ **DTBModel (DM):** Implements the DTB neural model // `initialize_model()`
✧ **dtb_step (DS):** Performs a single simulation step // `step()`
✧ **Vi:** Membrane potential // `Vi = tf.Variable(tf.random.uniform([num_neurons], minval=-70.0, maxval=-50.0), dtype=tf.float32)`
✧
✧ **ji_u:** Channel opening probability // `ji_u = np.random.uniform(0, 1, size=(self.n_samples, self.n_neurons, self.n_neurotransmitters))`
✧
✧ **Ci:** Neuronal capacitance // `Ci = tf.constant(1.0, dtype=tf.float32)`
✧ **gL_i:** Leak conductance // `gL_i = tf.constant(0.1, dtype=tf.float32)`
✧ **EL:** Equilibrium potential // `EL = tf.constant(-65.0, dtype=tf.float32)`
✧ **Eu:** Reversal potentials // `Eu = tf.constant([-70.0, 0.0, -80.0, -90.0], dtype=tf.float32)`
✧
✧ **gi_u:** Synaptic conductance // `gi_u = tf.Variable(tf.random.uniform([num_neurons, num_synapses]), dtype=tf.float32)`
✧
✧ **Iext:** External current // `Iext = np.random.uniform(-1e-9, 1e-9, size=(self.n_samples, self.n_neurons, 1))`
✧ **Tu:** Decay constant // `Tu = tf.constant(5.0, dtype=tf.float32)`
✧ **wu:** Synaptic weight // `wu = tf.Variable(tf.random.uniform([num_synapses]), dtype=tf.float32)`
✧
✧ **Isum:** Sum of all synaptic and external currents // `Isum = synaptic_currents + Iext`

DG function represents the creation of synthetic data with random values within specified ranges. DM function represents the initialization of the DTB model with TensorFlow variables and constants. DS function represents a single time step in the simulation, updating the membrane potential and channel states based on the DTB equations. DTB_Simulation represents the repeated execution of the DS function for a given amount of time.

(I. DTB synthetic-data-generator for input to TF-model as output for CV_Adapt)
(II. DTB TensorFlow neural-model
(III. DTB neural-net Simulator which creates Digital Twin Brain (DTB)

(III. = component_4 input tensors to CV_Adapt (component_5) )

```
### THE DTB pseudoCODE, IN 3 FILES ###
#### 022425   1029PM   ####
## I. Synthetic Data Generator (data_generator.py):
##Generates three main categories of synthetic neural data:

import numpy as np
import tensorflow as tf
```

```
class DataGenerator:
    def __init__(self, n_samples=100, n_neurons=5, n_neurotransmitters=4,
n_presynaptic_neurons=3):
        self.n_samples = n_samples
        self.n_neurons = n_neurons
        self.n_neurotransmitters = n_neurotransmitters
        self.n_presynaptic_neurons = n_presynaptic_neurons
```

```
    def generate_data(self):
        try:
            # Neuronal parameters
            C = np.random.uniform(1e-9, 1e-6, size=(self.n_samples, self.n_neurons, 1))
# Capacitance in Farads
           gL = np.random.uniform(1e-9, 1e-6, size=(self.n_samples, self.n_neurons, 1))
# Conductance in Siemens
           V = np.random.uniform(-70, -50, size=(self.n_samples, self.n_neurons, 1))
# Membrane potential in mV
           EL = np.random.uniform(-80, -60, size=(self.n_samples, self.n_neurons, 1))
 # Leak potential in mV
```

```
            # Synaptic parameters (ensuring proper ranges)
            Isyn = np.random.uniform(0, 1e-9, size=(self.n_samples, self.n_neurons,
self.n_neurotransmitters))  # Current in Amperes
            Iext = np.random.uniform(-1e-9, 1e-9, size=(self.n_samples,
self.n_neurons, 1))  # External current in Amperes
            # Channel dynamics (ensuring proper ranges)
            ji_u = np.random.uniform(0, 1, size=(self.n_samples, self.n_neurons,
self.n_neurotransmitters))  # Channel opening probability
            Tau_u = np.random.uniform(1e-3, 1e-2, size=(self.n_samples,
self.n_neurons, self.n_neurotransmitters))  # Decay time in seconds
            omega_u = np.random.uniform(0, 1, size=(self.n_samples, self.n_neurons,
self.n_neurotransmitters))  # Synaptic weight (normalized)
 tkm = np.random.uniform(0, 100, size=(self.n_samples, self.n_neurons,
self.n_presynaptic_neurons))  # Time in milliseconds
```

```
    # Verification logging
    print("\nParameter Ranges Verification:")

    print(f"Capacitance range: [{np.min(C):.2e}, {np.max(C):.2e}] F")

    print(f"Conductance range: [{np.min(gL):.2e}, {np.max(gL):.2e}] S")

    print(f"Membrane potential range: [{np.min(V):.2f}, {np.max(V):.2f}] mV")

    print(f"Leak potential range: [{np.min(EL):.2f}, {np.max(EL):.2f}] mV")

    print(f"Synaptic current range: [{np.min(Isyn):.2e}, {np.max(Isyn):.2e}] A")

    print(f"External current range: [{np.min(Iext):.2e}, {np.max(Iext):.2e}] A")

    print(f"Channel opening range: [{np.min(ji_u):.2%}, {np.max(ji_u):.2%}]")

    print(f"Decay time range: [{np.min(Tau_u):.2e}, {np.max(Tau_u):.2e}] s")

    print(f"Synaptic weight range: [{np.min(omega_u):.2f}, {np.max(omega_u):.2f}]")

    print(f"Spike timing range: [{np.min(tkm):.2f}, {np.max(tkm):.2f}] ms\n")
```

```
            # Combine all data
input_data = np.concatenate([C, gL, V, EL, Isyn, Iext, ji_u, Tau_u, omega_u, tkm],
axis=-1)
            return tf.convert_to_tensor(input_data, dtype=tf.float32)
```

```
        except Exception as e:
            raise Exception(f"Error generating synthetic data: {str(e)}")
```

```
    def get_data_labels(self):
        return [
            "Capacitance (C)",
            "Leak Conductance (gL)",
            "Membrane Potential (V)",
            "Equilibrium Potential (EL)",
            "Synaptic Current (Isyn)",
            "External Current (Iext)",
            "Channel Opening (ji_u)",
            "Decay Time Constant (Tau_u)",
            "Synaptic Weight (omega_u)",
            "Presynaptic Spike Timing (tkm)"




        ]
```

DTB synthetic-data-generator for input to TF-model as output for CV_Adapt)
DTB TensorFlow neural-model
DTB neural-net Simulator which creates Digital Twin Brain (DTB)

(III. = component_4 input tensors to CV_Adapt (component_5) )

Synthetic data DTB

```
2025-02-25 02:24:21,038 - INFO - Starting Neural Network Data Display
2025-02-25 02:24:21,039 - INFO - Python version: 3.11.10 (main, Sep  7 2024, 01:03:31) [GCC 13.3.0]
2025-02-25 02:24:21,039 - INFO - NumPy version: 2.0.0
2025-02-25 02:24:21,039 - INFO - TensorFlow version: 2.18.0
2025-02-25 02:24:21,039 - INFO - Initializing components...
2025-02-25 02:24:21,039 - INFO - Initializing ModelRunner with 5 neurons and 4 synapses
2025-02-25 02:24:21.040026: E external/local_xla/xla/stream_executor/cuda/cuda_driver.cc:152] failed call to cuInit: INTERNAL: CUDA error: Failed call to cuInit: UNKNOWN ERROR (303)
2025-02-25 02:24:21,061 - INFO - Model initialized with shapes:
2025-02-25 02:24:21,061 - INFO - Vi shape: (5,)
2025-02-25 02:24:21,061 - INFO - ji_u shape: (5, 4)
2025-02-25 02:24:21,061 - INFO - Eu shape: (4,)
2025-02-25 02:24:21,061 - INFO - Generating synthetic neural data...
2025-02-25 02:24:21,067 - INFO - Displaying input data summary...
================================================================
                        Input Data Statistics
================================================================
Capacitance (C)               | mean: 4.74e-07, std: 2.89e-07
Leak Conductance (gL)         | mean: 5.13e-07, std: 2.86e-07
Membrane Potential (V)        | mean: -6.00e+01, std: 5.79e+00
Equilibrium Potential (EL)    | mean: -6.98e+01, std: 5.58e+00
Synaptic Current (Isyn)       | mean: 4.68e-09, std: 3.20e-09
External Current (Iext)       | mean: 4.48e-09, std: 3.23e-09
Channel Opening (ji_u)        | mean: 4.49e-09, std: 3.21e-09
Decay Time Constant (Tau_u)   | mean: 4.62e-09, std: 3.19e-09
Synaptic Weight (omega_u)     | mean: 4.41e-09, std: 3.09e-09
Presynaptic Spike Timing (tkm) | mean: 4.98e-01, std: 2.94e-01
================================================================
2025-02-25 02:24:21,072 - INFO - Running model simulation...
2025-02-25 02:24:21,072 - INFO - Running simulation for 10 steps
================================================================
                        Model Execution Results
================================================================

Step 1:
----------------------------------------
Membrane Potentials    | mean: -6.01e+01, std: 6.18e+00
Channel States         | mean: 3.33e-03, std: 1.72e-03
Synaptic Currents      | mean: 0.00e+00, std: 0.00e+00

Step 2:
----------------------------------------
```

Synthetic data DTB inputs

```
Generating synthetic neural network data...

Parameter Ranges Verification:
Capacitance range: [2.81e-08, 9.98e-07] F
Conductance range: [1.15e-08, 9.96e-07] S
Membrane potential range: [-69.53, -51.58] mV
Leak potential range: [-79.91, -60.25] mV
Synaptic current range: [2.47e-12, 9.96e-10] A
External current range: [-9.97e-10, 9.75e-10] A
Channel opening range: [0.51%, 99.80%]
Decay time range: [1.03e-03, 9.99e-03] s
Synaptic weight range: [0.00, 0.99]
Spike timing range: [0.45, 99.26] ms

2025-02-25 03:00:15.227850: E external/local_xla/xla/stream_executor/cuda/cuda_driver.cc:152] f
================================================================================================
                            Neural Network Synthetic Data Statistics
        Synthetic-data inputs for TF-Model of DTB-equations output to CV_adapt (component_4)
                                    [ ai2agi by ai70000, Ltd ]
================================================================================================

Capacitance (C):
----------------------------------------
Minimum: 2.81e-08 F
Maximum: 9.98e-07 F
Mean: 4.95e-07 F
Standard Deviation: 2.55e-07 F

Leak Conductance (gL):
----------------------------------------
Minimum: 1.15e-08 S
Maximum: 9.96e-07 S
Mean: 4.67e-07 S
Standard Deviation: 2.83e-07 S

Membrane Potential (V):
----------------------------------------
Minimum: -6.95e+01 mV
Maximum: -5.16e+01 mV
Mean: -6.00e+01 mV
```

Synthetic data DTB inputs

```
================================================================================
                    Neural Network Synthetic Data Statistics
        Synthetic-data inputs for TF-Model of DTB-equations output to CV_adapt (component_4)
                              [ ai2agi by ai70000, Ltd ]
================================================================================

Capacitance (C):
----------------------------------------
Minimum: 2.81e-08 F
Maximum: 9.98e-07 F
Mean: 4.95e-07 F
Standard Deviation: 2.55e-07 F

Leak Conductance (gL):
----------------------------------------
Minimum: 1.15e-08 S
Maximum: 9.96e-07 S
Mean: 4.67e-07 S
Standard Deviation: 2.83e-07 S

Membrane Potential (V):
----------------------------------------
Minimum: -6.95e+01 mV
Maximum: -5.16e+01 mV
Mean: -6.00e+01 mV
Standard Deviation: 5.68e+00 mV

Equilibrium Potential (EL):
----------------------------------------
Minimum: -7.99e+01 mV
Maximum: -6.02e+01 mV
Mean: -7.06e+01 mV
Standard Deviation: 6.18e+00 mV

Synaptic Current (Isyn):
----------------------------------------
Minimum: 2.47e-12 A
Maximum: 9.96e-10 A
Mean: 4.80e-10 A
Standard Deviation: 2.81e-10 A

External Current (Iext):
----------------------------------------
Minimum: 1.04e-11 A
Maximum: 9.78e-10 A
Mean: 4.60e-10 A
Standard Deviation: 2.79e-10 A

Channel Opening (ji_u):
```

```
# II. DTB Model Implementation (dtb_model.py):

import tensorflow as tf


class DTBModel:
    def __init__(self, num_neurons=100, num_synapses=100, dt=0.01):
        self.num_neurons = num_neurons
        self.num_synapses = num_synapses
        self.dt = dt
        self.initialize_model()

    def initialize_model(self):
        try:
            # Initialize state variables
            self.Vi = tf.Variable(tf.random.uniform([self.num_neurons],
minval=-70.0, maxval=-50.0))
            self.ji_u = tf.Variable(tf.zeros([self.num_neurons,
self.num_synapses]))

            # Constants
            self.Ci = tf.constant(1.0, dtype=tf.float32)
            self.gL_i = tf.constant(0.1, dtype=tf.float32)
            self.EL = tf.constant(-65.0, dtype=tf.float32)
            self.Eu = tf.constant([-70.0, 0.0, -80.0, -90.0], dtype=tf.float32)
            self.gi_u = tf.Variable(tf.random.uniform([self.num_neurons,
self.num_synapses]))
            self.Iext = tf.Variable(tf.zeros([self.num_neurons]))
            self.Tu = tf.constant(5.0, dtype=tf.float32)
            self.wu = tf.Variable(tf.random.uniform([self.num_synapses]))

        except Exception as e:
            raise Exception(f"Error initializing DTB model: {str(e)}")

    def step(self):
        try:
            # Compute synaptic currents
            Vi_expanded = tf.expand_dims(self.Vi, axis=1)
            drive_force = Vi_expanded - self.Eu
            synaptic_currents = tf.reduce_sum(self.ji_u * drive_force * self.gi_u,
axis=1)
            Isum = synaptic_currents + self.Iext

            # Update membrane potential
            dVi_dt = (-self.gL_i * (self.Vi - self.EL) + Isum) / self.Ci
            Vi_new = self.Vi + self.dt * dVi_dt

            # Update channel dynamics
            dji_u_dt = (-1.0 / self.Tu) * self.ji_u + self.wu *
```

```
tf.exp(-self.dt)
            ji_u_new = self.ji_u + self.dt * dji_u_dt
            # Apply updates
            self.Vi.assign(Vi_new)
            self.ji_u.assign(ji_u_new)

            return {
                'membrane_potential': self.Vi.numpy(),
                'channel_states': self.ji_u.numpy(),
                'synaptic_currents': synaptic_currents.numpy(),
                'total_current': Isum.numpy()
            }

        except Exception as e:
            raise Exception(f"Error during DTB step: {str(e)}")
```

Synthetic data TensorFlow outputs

```
Membrane Potentials     | mean: -6.01e+01, std: 6.17e+00
Channel States          | mean: 9.98e-03, std: 5.16e-03
Synaptic Currents       | mean: 2.71e-01, std: 1.56e-01

Step 4:
----------------------------------------
Membrane Potentials     | mean: -6.01e+01, std: 6.16e+00
Channel States          | mean: 1.33e-02, std: 6.88e-03
Synaptic Currents       | mean: 4.05e-01, std: 2.33e-01

Step 5:
----------------------------------------
Membrane Potentials     | mean: -6.01e+01, std: 6.16e+00
Channel States          | mean: 1.66e-02, std: 8.59e-03
Synaptic Currents       | mean: 5.40e-01, std: 3.10e-01

Step 6:
----------------------------------------
Membrane Potentials     | mean: -6.01e+01, std: 6.16e+00
Channel States          | mean: 1.99e-02, std: 1.03e-02
Synaptic Currents       | mean: 6.74e-01, std: 3.87e-01

Step 7:
----------------------------------------
Membrane Potentials     | mean: -6.01e+01, std: 6.15e+00
Channel States          | mean: 2.32e-02, std: 1.20e-02
Synaptic Currents       | mean: 8.08e-01, std: 4.64e-01

Step 8:
----------------------------------------
Membrane Potentials     | mean: -6.01e+01, std: 6.15e+00
Channel States          | mean: 2.65e-02, std: 1.37e-02
Synaptic Currents       | mean: 9.42e-01, std: 5.41e-01

Step 9:
----------------------------------------
Membrane Potentials     | mean: -6.01e+01, std: 6.15e+00
Channel States          | mean: 2.98e-02, std: 1.54e-02
Synaptic Currents       | mean: 1.08e+00, std: 6.18e-01

Step 10:
----------------------------------------
Membrane Potentials     | mean: -6.01e+01, std: 6.15e+00
Channel States          | mean: 3.30e-02, std: 1.71e-02
Synaptic Currents       | mean: 1.21e+00, std: 6.95e-01
================================================================================
2025-02-25 02:24:21,097 - INFO - Processing completed successfully
```

Synthetic data inputs of DTB equations

```
2025-02-25 03:37:19,156 - INFO - Displaying input data summary...

Parameter Ranges Verification:
Capacitance (C) range: [1.8068e-09, 9.9952e-07]
Leak Conductance (gL) range: [1.3895e-09, 9.9957e-07]
Membrane Potential (V) range: [-6.9966e+01, -5.0075e+01]
Equilibrium Potential (EL) range: [-7.9981e+01, -6.0106e+01]
Synaptic Current (Isyn) range: [1.2816e-13, 9.9820e-10]
External Current (Iext) range: [5.5040e-12, 9.9881e-10]
Channel Opening (ji_u) range: [9.6677e-13, 9.9930e-10]
Decay Time Constant (Tau_u) range: [1.0485e-12, 9.9875e-10]
Synaptic Weight (omega_u) range: [-9.9963e-10, 9.9067e-10]
Presynaptic Spike Timing (tkm) range: [1.9996e-03, 9.9854e-01]

================================================================================================
2025-02-25 03:37:19,162 - INFO - Running model simulation...
2025-02-25 03:37:19,162 - INFO - Running simulation for 10 steps

Model Simulation Results:
================================================================================================
Synthetic-data-inputs of DTB-equations (component_5) to TF-Model for output as tensors to CV_adapt (c
                                [ ai2agi  by:  ai70000,Ltd. ]
                             Neural Network Model Simulation Output
================================================================================================

Step 1
----------------------------------------
Membrane Potentials (mV):
[-53.4689 -54.8964 -55.2473 -52.0447 -53.4097]

Channel States (%):
[[0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]]

Synaptic Currents (nA):
[0. 0. 0. 0. 0.]

Step 2
----------------------------------------
Membrane Potentials (mV):
[-53.4799 -54.9075 -55.2567 -52.0592 -53.4222]

Channel States (%):
[[1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
```

Synthetic data TensorFlow outputs

```
Step 2
----------------------------------------
Membrane Potentials (mV):
[-53.4799 -54.9075 -55.2567 -52.0592 -53.4222]

Channel States (%):
[[1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]]

Synaptic Currents (nA):
[ 0.0526 -0.0971  0.0332 -0.1545 -0.0948]

Step 3
----------------------------------------
Membrane Potentials (mV):
[-53.4904 -54.9196 -55.2658 -52.0752 -53.4357]

Channel States (%):
[[2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]]

Synaptic Currents (nA):
[ 0.1049 -0.1943  0.0661 -0.3092 -0.1896]

Step 4
----------------------------------------
Membrane Potentials (mV):
[-53.5003 -54.9326 -55.2746 -52.0928 -53.4501]

Channel States (%):
[[2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]]

Synaptic Currents (nA):
[ 0.157  -0.2918  0.0988 -0.464  -0.2844]

Step 5
----------------------------------------
Membrane Potentials (mV):
```

```
# III. Basic DTB Implementation (dtb_basic.py):

import tensorflow as tf
import numpy as np
from data_generator import DataGenerator

# Create synthetic data
data_gen = DataGenerator(n_samples=100, n_neurons=100)
input_data = data_gen.generate_data()

# Define DTB parameters as tensors
num_neurons = 100  # Example neuron count
num_synapses = 4   # Match with number of neurotransmitter types (AMPA, NMDA, GABAa,
GABAb)

dt = 0.01  # Time step

# Neuronal states
Vi = tf.Variable(tf.random.uniform([num_neurons], minval=-70.0, maxval=-50.0),
dtype=tf.float32)  # Membrane potentials
ji_u = tf.Variable(tf.zeros([num_neurons, num_synapses]), dtype=tf.float32)  # Open
channel percentages

# Constants
Ci = tf.constant(1.0, dtype=tf.float32)  # Neuronal capacitance
gL_i = tf.constant(0.1, dtype=tf.float32)  # Leak conductance
EL = tf.constant(-65.0, dtype=tf.float32)  # Equilibrium potential
Eu = tf.constant([-70.0, 0.0, -80.0, -90.0], dtype=tf.float32)  # Reversal potentials
for AMPA, NMDA, GABAa, GABAb
gi_u = tf.Variable(tf.random.uniform([num_neurons, num_synapses]),
dtype=tf.float32)  # Synaptic conductance
Iext = tf.Variable(tf.zeros([num_neurons]), dtype=tf.float32)  # External current
Tu = tf.constant(5.0, dtype=tf.float32)  # Decay constant
wu = tf.Variable(tf.random.uniform([num_synapses]), dtype=tf.float32)  # Synaptic
weight

def format_tensor(tensor):
    """Format tensor values to 3 decimal places"""
    if isinstance(tensor, tf.Tensor):
        tensor = tensor.numpy()
    return np.round(tensor, decimals=3)
```

```
def dtb_step():
    # Compute Isum (sum of synaptic and external currents)
    Vi_expanded = tf.expand_dims(Vi, axis=1)  # Expand for broadcasting
    drive_force = Vi_expanded - Eu  # Driving force
    synaptic_currents = tf.reduce_sum(ji_u * drive_force * gi_u, axis=1)  # Compute
sum of all synaptic currents
    Isum = synaptic_currents + Iext

    # Compute membrane potential update
    dVi_dt = (-gL_i * (Vi - EL) + Isum) / Ci
    Vi_new = Vi + dt * dVi_dt

    # Compute neurotransmitter channel dynamics
    dji_u_dt = (-1.0 / Tu) * ji_u + wu * tf.exp(-dt)  # Placeholder for spike influence
    ji_u_new = ji_u + dt * dji_u_dt

    # Apply updates
    Vi.assign(Vi_new)
    ji_u.assign(ji_u_new)

    return {
        'Vi': Vi_new,
        'dVi_dt': dVi_dt,
        'ji_u': ji_u_new,
        'dji_u_dt': dji_u_dt,
        'synaptic_currents': synaptic_currents,
        'Isum': Isum
    }

print("\nSynthetic Input Data Tensor Shape:", input_data.shape)
print("Synthetic Input Data (first few elements):", format_tensor(input_data[0,
0, :]))

print("\nStarting DTB simulation...")
for step in range(10):
    step_result = dtb_step()
    print(f"\nStep {step + 1} DTB Equation Results:")
    print("-" * 50)

    # Membrane Potential Equation: dVi/dt = (-gL_i * (Vi - EL) + Isum) / Ci
    print("1. Membrane Potential (dVi/dt):")
    print("   Vi =", format_tensor(step_result['Vi'][:5]))
    print("   dVi_dt =", format_tensor(step_result['dVi_dt'][:5]))
```

```
# Channel Dynamics Equation: dji_u/dt = -ji_u/Tu + wu * exp(-dt)
    print("\n2. Channel Dynamics (dji_u/dt):")
    print("   ji_u =", format_tensor(step_result['ji_u'][:2, :2]))
    print("   dji_u_dt =",

format_tensor(step_result['dji_u_dt'][:2, :2]))
```

```
    # Synaptic Current Equation: Isyn = Σ(ji_u * (Vi - Eu) * gi_u)
    print("\n3. Synaptic Currents (Isyn):")
    print("   synaptic_currents =",
format_tensor(step_result['synaptic_currents'][:5]))
    print("   Total current (Isum) =", format_tensor(step_result['Isum'][:5]))
```

```
print("\nDTB step execution complete.")
```

inputs

```
2025-02-25 03:37:19,156 - INFO - Displaying input data summary...

Parameter Ranges Verification:
Capacitance (C) range: [1.8068e-09, 9.9952e-07]
Leak Conductance (gL) range: [1.3895e-09, 9.9957e-07]
Membrane Potential (V) range: [-6.9966e+01, -5.0075e+01]
Equilibrium Potential (EL) range: [-7.9981e+01, -6.0106e+01]
Synaptic Current (Isyn) range: [1.2816e-13, 9.9820e-10]
External Current (Iext) range: [5.5040e-12, 9.9881e-10]
Channel Opening (ji_u) range: [9.6677e-13, 9.9930e-10]
Decay Time Constant (Tau_u) range: [1.0485e-12, 9.9875e-10]
Synaptic Weight (omega_u) range: [-9.9963e-10, 9.9067e-10]
Presynaptic Spike Timing (tkm) range: [1.9996e-03, 9.9854e-01]

================================================================================
2025-02-25 03:37:19,162 - INFO - Running model simulation...
2025-02-25 03:37:19,162 - INFO - Running simulation for 10 steps

Model Simulation Results:
================================================================================
Synthetic-data-inputs of DTB-equations (component_5) to TF-Model for output as tensors to CV_adapt (c
                                [ ai2agi  by:  ai70000,Ltd. ]
                            Neural Network Model Simulation Output
================================================================================

Step 1
----------------------------------------
Membrane Potentials (mV):
[-53.4689 -54.8964 -55.2473 -52.0447 -53.4097]

Channel States (%):
[[0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]
 [0.7083 0.7658 0.1859 0.5266]]

Synaptic Currents (nA):
[0. 0. 0. 0. 0.]

Step 2
----------------------------------------
Membrane Potentials (mV):
[-53.4799 -54.9075 -55.2567 -52.0592 -53.4222]

Channel States (%):
[[1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
```

outputs

```
Step 2
--------------------------------------------
Membrane Potentials (mV):
[-53.4799 -54.9075 -55.2567 -52.0592 -53.4222]

Channel States (%):
[[1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]
 [1.4153 1.5302 0.3715 1.0522]]

Synaptic Currents (nA):
[ 0.0526 -0.0971  0.0332 -0.1545 -0.0948]

Step 3
--------------------------------------------
Membrane Potentials (mV):
[-53.4904 -54.9196 -55.2658 -52.0752 -53.4357]

Channel States (%):
[[2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]
 [2.1208 2.2929 0.5567 1.5767]]

Synaptic Currents (nA):
[ 0.1049 -0.1943  0.0661 -0.3092 -0.1896]

Step 4
--------------------------------------------
Membrane Potentials (mV):
[-53.5003 -54.9326 -55.2746 -52.0928 -53.4501]

Channel States (%):
[[2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]
 [2.8249 3.0542 0.7415 2.1002]]

Synaptic Currents (nA):
[ 0.157  -0.2918  0.0988 -0.464  -0.2844]

Step 5
--------------------------------------------
Membrane Potentials (mV):
```

outputs

```
Step 6
----------------------------------------
Membrane Potentials (mV):
[-53.5186 -54.9614 -55.2911 -52.1325 -53.4817]

Channel States (%):
[[4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]]

Synaptic Currents (nA):
[ 0.2604 -0.4876  0.1634 -0.7746 -0.4742]

Step 7
----------------------------------------
Membrane Potentials (mV):
[-53.527  -54.9773 -55.2988 -52.1547 -53.4989]

Channel States (%):
[[4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]]

Synaptic Currents (nA):
[ 0.3118 -0.586   0.1953 -0.9304 -0.5693]

Step 8
----------------------------------------
Membrane Potentials (mV):
[-53.5348 -54.9942 -55.3063 -52.1784 -53.5171]

Channel States (%):
[[5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]]

Synaptic Currents (nA):
[ 0.363  -0.6848  0.2271 -1.0867 -0.6644]

Step 9
----------------------------------------
Membrane Potentials (mV):
```

outputs

```
Step 6
----------------------------------------
Membrane Potentials (mV):
[-53.5186 -54.9614 -55.2911 -52.1325 -53.4817]

Channel States (%):
[[4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]]

Synaptic Currents (nA):
[ 0.2604 -0.4876  0.1634 -0.7746 -0.4742]

Step 7
----------------------------------------
Membrane Potentials (mV):
[-53.527  -54.9773 -55.2988 -52.1547 -53.4989]

Channel States (%):
[[4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]]

Synaptic Currents (nA):
[ 0.3118 -0.586   0.1953 -0.9304 -0.5693]

Step 8
----------------------------------------
Membrane Potentials (mV):
[-53.5348 -54.9942 -55.3063 -52.1784 -53.5171]

Channel States (%):
[[5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]]

Synaptic Currents (nA):
[ 0.363  -0.6848  0.2271 -1.0867 -0.6644]

Step 9
----------------------------------------
Membrane Potentials (mV):
```

outputs

```
Step 6
----------------------------------------
Membrane Potentials (mV):
[-53.5186 -54.9614 -55.2911 -52.1325 -53.4817]

Channel States (%):
[[4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]
 [4.2289 4.5721 1.11   3.144 ]]

Synaptic Currents (nA):
[ 0.2604 -0.4876  0.1634 -0.7746 -0.4742]

Step 7
----------------------------------------
Membrane Potentials (mV):
[-53.527  -54.9773 -55.2988 -52.1547 -53.4989]

Channel States (%):
[[4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]
 [4.9288 5.3288 1.2937 3.6643]]

Synaptic Currents (nA):
[ 0.3118 -0.586   0.1953 -0.9304 -0.5693]

Step 8
----------------------------------------
Membrane Potentials (mV):
[-53.5348 -54.9942 -55.3063 -52.1784 -53.5171]

Channel States (%):
[[5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]
 [5.6273 6.084  1.4771 4.1836]]

Synaptic Currents (nA):
[ 0.363  -0.6848  0.2271 -1.0867 -0.6644]

Step 9
----------------------------------------
Membrane Potentials (mV):
```

## CV_Adapt

**CV_Adapt and e_Generalization / e_Reasoning integrate with CV_Adapt** acting as the central processing unit, integrating tensor outputs from these other components. This processes and interprets MultiModal information, enabling the AGI to understand context and relationships. It generates output tensors that serve as the "DNA" for AGI, facilitating adaptable and intelligent responses.

Computer vision, an essential technology in AI, continues to evolve through methods such as convolutional neural networks (CNNs) and deep learning models. Recent research demonstrates the growing sophistication of CV systems, from simple object recognition to more complex applications involving context-aware processing, multi-modal data fusion, and real-time adaptability in dynamic environments (LeCun, 2023) [109] (He et al., 2024) [24]. Recent developments focus on optimizing network architectures and reducing computation time while increasing accuracy, leveraging techniques such as transfer learning, self-supervised learning, and transformer models (Zhou et al., 2023) [30].

In Xie et al. (2023) its noted that visual signals are more complex than language signals, and that unification is the next important goal for computer-vision (cv) while at the same time noting computer-vision lacks the ability to learn form environments [32]. However, since 2023, modern computer vision can indeed learn from environments using self-supervised learning, reinforcement learning, and embodied AI. Models like Vision Transformers (ViTs) and diffusion models process environmental data, while robotics integrates vision with real-world interaction [33] [34] [35] [36].

Xie et al. (2023) [31] identifies open-world-recognition, the segment-anything task, generalized visual encoding, LLM-guided visual understanding, and MultiModal dialog as the main research topics towards a unification in computer-vision comparable to the progress witnessed in NLP [32].

And reading and interpreting tensors—multi-dimensional arrays used to store data in deep learning models—forms the core of the CV_Adapt functionality. Tensors, crucial in both training and inference phases of neural networks, allow the representation of data in a way that preserves relationships between features across multiple dimensions (Goodfellow et al., 2023) [28]. The tensors processed by CV_Adapt involve not only raw pixel data but also complex features related to the synaptic and neuronal dynamics from DTB, requiring sophisticated tensor operations to manipulate and extract meaningful insights.

(Voss & Jovanovic (2023)) [1] state AGI will will require means of capturing and interacting with the 4D world, perhaps via pc-screen, and be excellent tools users. Of note, it was Voss, along with Ben Goertzel and Shane Legg, who, in 2001 (re) coined the term ‘Artificial General Intelligence.’ [36][37]. The author(s) go onto differentiate ‘Thinking Machines’ (AGI) from ‘Narrow AI’ (AI.) and highlight the large-scale well-funded progress of DeepMind. The authors argue the original version of AI was to create these thinking machines to learn, reason and solve like humans. But instead the field has shifted to a focus on narrow ai, which can only solve one problem at a time [37]. The authors advocate for ‘Cognitive AI’ which is real-time, life-long conceptual learning and reasoning, which represents a direct path to AGI, which ultimately will solve the most difficult and pressing problems known to man.

Other state-of-the-art techniques in tensor-based computer vision include advancements in convolutional operations and the deployment of neural network models capable of handling high-dimensional tensor structures (Chen et al., 2024) [27]. These same techniques allow CV_Adapt to leverage high-throughput tensor computations, ensuring real-time performance when processing large amounts of visual and neural data simultaneously.

Another computer vision approach is examined in (Xie et al. (2023)) [31] with their algorithm based on learning from environments in three stages; establishing, pre-train, and, fine-tune [43], with the key difference from NLP being the training environments must be able to be interacted with. The authors state current computer-vision algorithms are too specialized and can't generalize well.

CV_Adapt, in part helps to overcome this issue. Furthermore, the CV_Adapt component remains robust enough to handle edge cases where the data doesn't conform to expected patterns, something particularly relevant when integrating the DTB outputs which simulate biological processes that are more prone to variability compared to standard computer vision inputs.

On a large enterprise level, DeepMind has made significant strides toward Artificial General Intelligence (AGI,) via introduction of Gato, a generalist agent capable of performing 600 diverse tasks across various domains. To systematically assess progress toward AGI, DeepMind proposed levels of AGI performance, generality, and autonomy. And recently (DeepMind 2025)) [38] formed a specialized AI research team focused on developing "world models" to simulate physical environments, aiming to enhance AI's understanding and interaction with the real world. DeepMind emphasized that it anticipates AGI is five to ten years away, a timeframe which aligns with the estimation used in this research project (dating back to 2023,) and confirmed as of the date of this research project paper [38]. By way of comparison the actual implementation of Gato is compared to this theoretical model ai2agi, in **Appendix 3**.

The CV_Adapt component_5 serves as the HPC cluster-manager (as GPU,) for all tensor inputs in the multi-faceted AI architecture that includes the neural network outputs from components_1, _2, _3, and _4. It operates as the key element for synthesizing and processing tensor data related to computer vision (CV), neuronal behavior, and digital twin brain (DTB) systems. The design of CV_Adapt reflects the integration of multiple sources of data, aligning with the current state-of-the-art approaches in computer vision, as well as recent advancements in neural modeling and tensor processing.

In partial support of this path, a potential new approach is proposed (Liu et al. (2023)) [44] to computer vision (CV) with natural language processing (NLP), especially the GPT series, which notes that since CV lacks a paradigm allowing for learning from environments, can instead use an imaginary pipeline [44]. The authors describe an epic odyssey towards AGI and define an algorithm replicating any intellectual human task. They also discuss research in CV focusing on open-world visual recognition, the segment anything task, generalized visual encoding, LLM-guided visual understanding, and MultiModal dialog. The authors propose an imaginary pipeline towards AGI in CV, including a set of environments that are; fidelitous, abundant, and interactable via algorithms; CLIP, ALIGN, SAM, pix2seq, and others [44].

At the intersection of computer vision and how they interpret images and videos is the Transformer model (Vaswani et al., 2017) [101], at times, enhanced by the JAMBA architecture, (Ai2I Labs (2024) . Initially created to excel at sequential (text) data for NLP, the 'attention mechanism' allows the model to weight the inputs. This mechanism was applied to image 'patches,' which led to Vision Transformers (ViTs) to process patch-embeddings instead of pixels, capturing long-range dependencies and global context (Dosovitskiy et al., 2020) [98]. So with CV_Adapt, image classification, object detection, semantics, and multimodal learning are processed on the inputs from components 1, 2, 3, and 4, leading to AGI being trained on video, audio, memories, and mass quantities of data to produce an olfactory sense, etc., akin to human learning. Endeavors on this scale can presently only be funded by OpenAI, Nvidia, Google DeepMind, Anthropic, and xAI. However, with CV_Adapt as a potential blueprint towards the next quantum leap from AI to AGI, and with integration with the DTB and the other three components, it is within the realm of possibility to take the next substantial advancement in the quest for true AGI.

Computer Vision and AGI via Neuro-Symbolic VQA is explored in (Berlot-Attwell, 2021) [9] at the intersection of computer vision and AGI through the lens of Visual Question Answering (VQA). It reviews neuro-symbolic models (combining neural nets with symbolic reasoning) as a contributor toward the creation of AGI via: reasoning about visual content as the core of CV, answering questions about images with both visual understanding and reasoning abilities (bridging the gap between CV and NLP), learning and generalization from limited data via neuro-symbolic models for better generalization on less training data, compositional generalization to understand new combinations of known concepts, and improved interpretability and explainability via insight into their reasoning, for transparency and trust (Berlot-Attwell, 2021 [9]).

So each of these research papers in its own way points to the somewhat overlooked potential of implementing CV as an integral component of true AGI, not from the literal sense of detecting physical objects akin to self-driving vehicles, but as the actual central processing unit (via GPU) and as the final system which outputs the AGI DNA.

In this context, CV_Adapt stands as a component that bridges the gap between the computational output (DTB) and traditional computer vision pipelines. It achieves this by interpreting and refining the multimodal tensor data that spans visual, sensory, and neural information. The need to adapt to varying tensor structures, including those derived from DTB's neuronal dynamics and the raw image data processed by CV systems, emphasizes the flexibility and complexity required in CV_Adapt's design.

To accomplish this, CV_Adapt integrates various tensor operations such as reshaping, element-wise multiplication, aggregation, and matrix transformations, which align with the output requirements of each input component. The DTB component feeds dynamic neural data into CV_Adapt, where it's normalized, filtered, and combined with visual data for downstream processing. One key aspect of the CV_Adapt system is its ability to process large, heterogeneous tensors effectively, ensuring that the integration of information from disparate sources (neuronal, visual, and sensory) can occur seamlessly, with minimal computational overhead. For an example of this refer to the synthetic-data-generator, its outputs and how the TensorFlow model outputs the tensors as inputs to CV_Adapt."

In conclusion for CV_Adapt, it's designed as a modular integral component within a larger architecture, addressing multimodal data and tensor processing. As computer vision technologies advance, the role of tensor operations in enabling accurate, efficient processing of complex visual and neural information continues to grow (LeCun et al., 2024) [132]. CV_Adapt plays a role in the next evolution of AI systems designed to achieve true artificial general intelligence (AGI), particularly in integrating diverse data streams (Baltrušaitis et al., 2018) [103].

To integrate the DTB component into the existing CV_Adapt pipeline, you combine the synthetic data generation and TensorFlow model from DTB with the tensor processing in CV_Adapt. The DTB component generates tensors that serve as one of the inputs to CV_Adapt, alongside the tensors from the other components (G_enhanced, R_enhanced, and ai_LLM). This integration reflects a trend toward heterogeneous computing in AI, where specialized modules contribute to overall system intelligence (Dean et al., 2012) [108].

What follows is the CV_Adapt pseudo-code, including the DTB component.

Based on the aforementioned description and summary, the pseudo-code (below) for component 5 (CV_Adapt) of true AGI is presented as the "central processor" (via GPU), which takes as input the tensor outputs from components 1, 2, 3, and 4, implemented in TensorFlow for tensor generation."

***Conclusion: output-tensors from CV_Adapt = AGI - dna***

**Cv-adapt**

**Key Concepts:**

- CV_Adapt: Central processing unit for tensor integration.
- Tensor Processing: Operations on multi-dimensional arrays.
- MultiModal Data: Integration of various data types (visual, neural).
- DTB Output: Tensor data from the Digital Twin Brain.
- CNNs/ViTs/Diffusion Models: Computer vision models.
- AGI-DNA: Output tensors representing AGI state.

Mathematical Notation:

✓ CV_Adapt as a Function:

- **CV_Adapt (T_G, T_R, T_LLM, T_DTB) → T_AGI**
- Where:

  **T_G, T_R, T_LLM, T_DTB =**

  **input tensors from G_enhanced, R_enhanced, ai_LLM, and DTB respectively.**

  **T_AGI = output tensor representing AGI-DNA.**

✓ Tensor Operations:

- **T_AGI = F( T_G, T_R, T_LLM, T_DTB )**
- Where:

  **F = function; tensor operations (reshape, element-wise multiplication, aggregation, matrix transformations).**

✓ MultiModal Integration:

- **T_AGI = Integrate(T_Visual, T_Neural)**
- Where:

  **T_Visual = visual data tensors,**

  **T_Neural = neural data tensors (from DTB),**

  **Integrate = function: visual + neural tensors.**

✓ Computer Vision Models:

- **CV_Model(T_Input) → T_Features**
- **Where:**

  **CV_Model = CNNs, ViTs, or diffusion models**

  **T_Input = input tensor (e.g., image patches),**

  **T_Features = output feature tensor.**

- ✓ Environmental Learning:
  - **T_Features = Learn(T_Environment)**
  - Where:

    **Learn = self-supervised learning, reinforcement learning, or embodied AI. T_Environment = tensor representing environmental data.**

- ✓ AGI Progression: **AGI = CV_Adapt(G_enhanced, R_enhanced, ai_LLM, DTB)**

Start:

- **CV_Adapt(T_G, T_R, T_LLM, T_DTB) → T_AGI**
- **T_AGI = F(T_G, T_R, T_LLM, T_DTB)**
- **T_AGI = Integrate(T_Visual, T_Neural)**

Mathematical Notation:

- ✓ Function Decomposition:
  - **T_AGI = F( T_G, T_R, T_LLM, T_DTB )** is decomposed into sequential operations:
    - Receive(T_g, T_r, T_ai, T_dtb) → Input Tensors
    - Preprocess(Input Tensors) → Normalized Tensors
    - Fuse(Normalized Tensors) → Fused Tensor
    - Adapt(Fused Tensor) → Adapted Tensor
    - Store(Adapted Tensor) → Stored Tensor
    - Output(Stored Tensor) → T_AGI

- ✓ Mapping Text Concepts to Code Functions:
  - **F ( T_G, T_R, T_LLM, T_DTB )** maps to the sequence: Fuse(Preprocess(Receive(...)))
  - Integrate(T_Visual, T_Neural) maps to the fusion process, where T_Visual, T_Neural are subsets of the input tensors.
  - Central processing maps to Adapt() function.

- ✓ Explicit Memory and Output:
  - Code explicitly includes memory (Store) and output (Output) steps, which are implicit in the text's abstract function F.

- ✓ Normalization:
  - The code explicitly normalizes the tensors, through the Preprocess() function, which is an implicit part of: F.

Mathematical Notation:

- ✓ Input Reception:
  - Receive **(T_g, T_r, T_ai, T_dtb) → {T_g, T_r, T_ai, T_dtb}**

- ✓ Tensor Preprocessing:
  - Preprocess(**{T_g, T_r, T_ai, T_dtb}**) → {Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb)}

- ✓ Tensor Fusion:
  - Fuse( **{ Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb) }** ) → Concatenate ( **Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb** ) )
- ✓ Tensor Adaptation**:**
  - Adapt (Concatenate (**Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb))) →**

    **Apply_Central_Processing(Concatenate(Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb)))**
- ✓ Memory Storage:
  - **Store(Apply_Central_Processing(Concatenate(Normalize(T_g),Normalize(T_r), Normalize(T_ai), Normalize(T_dtb)))) →**

    **Update_Memory(Apply_Central_Processing(Concatenate(Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb))))**
- ✓ Final Output:
  - **Output(Update_Memory(Apply_Central_Processing(Concatenate(Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb))))) →**

    **Apply_Central_Processing(Concatenate(Normalize(T_g), Normalize(T_r), Normalize(T_ai), Normalize(T_dtb)))**
- ✓ CV_Adapt Pipeline:
  - **CV_Adapt(T_g, T_r, T_ai, T_dtb) →**

    **Output(Store(Adapt(Fuse(Preprocess(Receive(T_g, T_r, T_ai, T_dtb))))))**
- ✓ DTB Tensor Generation:
  - **Generate_DTB() → Tensor(C, gL, V, EL, Isyn, Iext, ji_u, Tau_u, omega_u, tkm)**

✓ End (Code-Derived):

➢ **CV_Adapt(T_g, T_r, T_ai, T_dtb) →**

**Output( Store ( Adapt( Fuse( Preprocess( Receive ( T_g, T_r, T_ai, T_dtb ) )))))**

**Where:**

✧ **CV_Adapt:** Main processing pipeline // `initialize_cv_adapt_system()`
✧ **Receive(T_g, T_r, T_ai, T_dtb):** Receives input tensors // `receive_tensor_inputs()`
✧ **Preprocess(Tensors):** Normalizes input tensors // `preprocess_tensors()`
✧ **Fuse(Tensors):** Combines normalized tensors // `fuse_tensors()`
✧ **Adapt(T):** Applies central processing logic // `adapt_fused_tensor()`
✧ **Store(T):** Stores adapted tensor in memory // `tore_tensor_in_memory()`
✧ **Output(T):** Returns final processed tensor // `cv_adapt_final_output()`
✧ **Generate_DTB():** Generates synthetic DTB tensor // `generate_dtb_tensor()`
✧ **Normalize(T):** Normalizes a single tensor // `preprocess_tensors()`

✧ **Concatenate(T1, T2, T3, T4):** Concatenates tensors // `np.concatenate([C, gL, V, EL, Isyn, Iext, ji_u, Tau_u, omega_u, tkm], axis=-1)`
✧ **Apply_Central_Processing(T):** Applies central processing // `cv_adapt_pipeline(tensor_g, tensor_r, tensor_ai, tensor_dtb)`
✧ **Update_Memory(T):** Updates memory // `update_memory(adapted_tensor)`

```
# Pseudo-code for CV_adapt: Central Processor for Tensor Inputs from all modules including DTB

# Initialize CV_adapt System
initialize_cv_adapt_system()  # Initialize central processor for tensor data handling
initialize_central_memory()  # Initialize shared memory to store integrated tensor data
initialize_processing_pipeline()  # Initialize pipeline for tensor processing and adaptation

# Input Data Handling: Tensor Inputs from G_enhanced, R_enhanced, AI_LLM, and DTB
def receive_tensor_inputs(tensor_g, tensor_r, tensor_ai, tensor_dtb):
    # Collect tensor inputs from the three modules and DTB
    all_tensors = {
        'tensor_g': tensor_g,
        'tensor_r': tensor_r,
        'tensor_ai': tensor_ai,
        'tensor_dtb': tensor_dtb  # Add DTB tensor
    }
    return all_tensors

# Tensor Normalization and Preprocessing: Standardize tensor inputs
def preprocess_tensors(tensors):
    # Normalize tensor data from all modules to ensure compatibility
    normalized_tensors = {
        'tensor_g': normalize_tensor(tensors['tensor_g']),
        'tensor_r': normalize_tensor(tensors['tensor_r']),
        'tensor_ai': normalize_tensor(tensors['tensor_ai']),
        'tensor_dtb': normalize_tensor(tensors['tensor_dtb'])  # Normalize DTB tensor
    }
    return normalized_tensors

# Tensor Fusion: Combine tensors into a unified representation
def fuse_tensors(normalized_tensors):
    # Fuse tensors using a fusion technique (e.g., concatenation, attention, etc.)
    fused_tensor = concatenate_tensors(normalized_tensors['tensor_g'],
                                       normalized_tensors['tensor_r'],
                                       normalized_tensors['tensor_ai'],
                                       normalized_tensors['tensor_dtb'])  # Include DTB tensor
    return fused_tensor
```

```
# Tensor Adaptation: Apply adaptation logic to process the fused tensor
def adapt_fused_tensor(fused_tensor):
    # Apply central processing logic to adapt the tensor for further use
    adapted_tensor = apply_central_processing(fused_tensor)
    return adapted_tensor

# Memory Integration: Store the adapted tensor into the shared memory for future use
def store_tensor_in_memory(adapted_tensor):
    # Store the adapted tensor in the central memory for later retrieval
    update_memory(adapted_tensor)
    return adapted_tensor

# Tensor Output: Final output of the central processor for further use
def cv_adapt_final_output(adapted_tensor):
    # Return the final processed tensor for use in other AGI components
    return adapted_tensor

# Central Tensor Processing Pipeline: Process inputs through CV_adapt
def cv_adapt_pipeline(tensor_g, tensor_r, tensor_ai, tensor_dtb):
    # Step 1: Receive tensor inputs from the three modules and DTB
    tensors = receive_tensor_inputs(tensor_g, tensor_r, tensor_ai, tensor_dtb)


# Step 2: Preprocess the tensors (normalization)
    normalized_tensors = preprocess_tensors(tensors)

    # Step 3: Fuse the tensors into a unified representation
    fused_tensor = fuse_tensors(normalized_tensors)

    # Step 4: Adapt the fused tensor (central processing)
    adapted_tensor = adapt_fused_tensor(fused_tensor)

    # Step 5: Store the adapted tensor in memory for future use
    final_tensor = store_tensor_in_memory(adapted_tensor)

    # Step 6: Output the final tensor for further AGI processing
    return cv_adapt_final_output(final_tensor)
```

```
# Generate synthetic data from DTB (Component 5)
def generate_dtb_tensor():
    # Generate synthetic data as per the DTB model (previously outlined in the DTB
component)
    n_samples = 100
    n_neurons = 5
    n_neurotransmitters = 4
    n_presynaptic_neurons = 3
    # Random synthetic data for each of the neuron dynamics (based on the DTB equations)
    C = np.random.uniform(1e-9, 1e-6, size=(n_samples, n_neurons, 1))
    gL = np.random.uniform(1e-9, 1e-6, size=(n_samples, n_neurons, 1))
    V = np.random.uniform(-70, -50, size=(n_samples, n_neurons, 1))
    EL = np.random.uniform(-80, -60, size=(n_samples, n_neurons, 1))
    Isyn = np.random.uniform(-1e-9, 1e-8, size=(n_samples, n_neurons,
n_neurotransmitters))
    Iext = np.random.uniform(-1e-9, 1e-8, size=(n_samples, n_neurons, 1))
    ji_u = np.random.uniform(0, 1, size=(n_samples, n_neurons, n_neurotransmitters))
    Tau_u = np.random.uniform(1e-3, 1e-2, size=(n_samples, n_neurons,
n_neurotransmitters))
    omega_u = np.random.uniform(1e-3, 1e-2, size=(n_samples, n_neurons,
n_neurotransmitters))
    tkm = np.random.uniform(0, 100, size=(n_samples, n_neurons,
n_presynaptic_neurons))

    # Stack all variables into a final input tensor
    input_data = np.concatenate([C, gL, V, EL, Isyn, Iext, ji_u, Tau_u, omega_u, tkm],
axis=-1)

    # Convert to TensorFlow tensor
    input_tensor = tf.convert_to_tensor(input_data, dtype=tf.float32)

    return input_tensor

# Example usage: Passing tensors from G_enhanced, R_enhanced, AI_LLM, and DTB
tensor_g = generate_tensor_from_G_enhanced()  # Output tensor from G_enhanced module
tensor_r = generate_tensor_from_R_enhanced()  # Output tensor from R_enhanced module
tensor_ai = generate_tensor_from_AI_LLM()  # Output tensor from AI_LLM module
tensor_dtb = generate_dtb_tensor()  # Output tensor from DTB module

# Process tensors through CV_adapt
final_output_tensor = cv_adapt_pipeline(tensor_g, tensor_r, tensor_ai, tensor_dtb)

# Further processing of final_output_tensor can now be done by other components of
AGI
```

**CV_Adapt s**ummary: :

- **DTB Synthetic Data**: The generate_dtb_tensor() function creates synthetic data based on the equations and variables described in the DTB component. This data is generated and converted into a TensorFlow tensor for integration.
- **Integration into CV_adapt**: The tensor generated by the DTB component (tensor_dtb) is now treated as one of the inputs to **CV_adapt**. It follows the same preprocessing, normalization, fusion, adaptation, and memory storage pipeline as the other tensors.
- **Fusion**: In the fuse_tensors() function, the DTB tensor is fused with the tensors from G_enhanced, R_enhanced, and AI_LLM to create a unified tensor representation.
- **Processing Pipeline**: The updated pipeline (cv_adapt_pipeline) ensures that the DTB tensor is integrated smoothly into the data flow alongside other module outputs.
- **The final output of this pseudo-code (once hard-coded source-coded,) is adapted into TensorFlow to generate tensors, used as input to component_5 CV_adapt.**

## CV_Adapt explanation / pseudocode

CV_adapt is the central processor (via GPU) for the AGI architecture. It integrates and processes tensor inputs from the four components: G_enhanced, R_enhanced, ai_LLM, and DTB, and then produces a final output tensor.

CV_adapt is initialized with functions to set up the central processor, shared memory, and processing pipeline. The `receive_tensor_inputs` function collects the four input tensors into a dictionary. The `preprocess_tensors` function normalizes these tensors to ensure compatibly. The `fuse_tensors` function combines the normalized tensors into a unified representation, using concatenation.

The `adapt_fused_tensor` function applies central processing logic to the fused tensor, adapting it for further use. The `store_tensor_in_memory` function stores the adapted tensor in shared memory for future retrieval. The `cv_adapt_final_output` function returns the final processed tensor.

The `cv_adapt_pipeline` function orchestrates the process, calling other functions in sequence. It receives four input tensors, preprocesses them, fuses them, adapts them, stores them in memory, and then returns the final output tensor.

A `generate_dtb_tensor` function is included to create synthetic data for DTB, mirroring the logic used in DTB module itself. Example usage is provided to demonstrate how CV_adapt pipeline is called with input tensors from the other components. The final output tensor can also be used by other parts of the AGI system, replicating the AGI dna.

## Conclusion

ai_LLMs and large-language-models are one of the greatest inventions in world-history. Their impact on the world of technology and humanity cannot be overstated.

That said, AI as it exists as of the date of this research project will not be the main component, nor the only component to achieve true AGI. Concurrently, AI / ai_LLM's *will* be *one* of the components of true AGI.

This paper proposes a possible, incremental advancement towards a potential foundation for true AGI, and that formula is:

**AGI ≈ f ( G_enhanced , R_enhanced, ai_LLM, DTB, CV_Adapt )**

Each component has been detailed in this research project paper, commenced in 2023 prior to the proliferation and extraordinary success and impact of the current ai_LLMs, and at the time focused primarily on NLP, and includes pseudo-code as an introductory starting-point blueprint for the infrastructure of true AGI within the next 7 years. It's feasible AGI may never happen, or, if it does, will exist in form completely different than this research project. But based on the totality of the current state of AI and the other technologies referenced, this proposal presents one of many possible, incremental steps forward.

For the first three components, G_enhanced , R_enhanced, and ai_LLM, almost all the schematics and approaches have already been extensively researched in recent years, and then implemented in public, private and agency domains.

However, the combination of those components, in conjunction with DTB, and o/i with CV_Adapt, using computer-vision as the 'central processor' for the AGI, is a macro level differentiator from other proposed methods, as of the date of this research project paper.

For component_4 DTB, due to time and resource constraints, a synthetic-data-generator was instantiated to generate data sufficiently proximate to real-world data the AGI would receive and ultimately use for inference for the process of applying the model and system for generalized intelligence, understanding, learning, and applying knowledge across many tasks.

Unlike ai_LLMs which are designed for specific tasks, true AGI inference involves the system making decisions and solving problems in ways similar to human intelligence, with the ability to transfer knowledge from one domain to another.

This schematic and pseudo-code represent one of many potential paths which might present an approach towards possible future research and exploration to develop true AGI, of which AI / ai_LLMs will be an integral, but not the only component.

Regardless of if or when true AGI is achieved, the present capabilities of ai_LLMs represent a tectonic shift in the world of technology and for all of humanity, and their power and potential cannot be overstated.

## Appendix 1

- **DL** – Deep Learning
- **ANN** – Artificial Neural Network
- **CV** – Computer Vision
- **R/IA** – Robotics / Intelligent Agents
- **NLP** – Natural Language Processing
- **SR/S** – Speech Recognition / Synthesis
- **ES** – Expert Systems
- **KRR** – Knowledge Representation and Reasoning
- **PSO** – Particle Swarm Optimization
- **AR/VR** – Augmented Reality / Virtual Reality

## Appendix 2: Overview of AGI components

**1. Dynamic Input**: e_Generalization and e_Reasoning components provide high-level, abstracted tensors representing learned patterns and logical structures, fed to CV_Adapt for generalized reasoning and deep logical inference.

The ai_LLM outputs are used intermittently, providing context and large-scale linguistic and cognitive insights, fed into CV_Adapt when language understanding or knowledge retrieval is required.

The DTB (Digital Twin) provides a real-time, detailed simulation model of the system or environment for dynamic state representations.

Flexible Input: Group 1, e_Generalization and e_Reasoning, feed together when high-level reasoning and pattern recognition are needed for complex tasks. Group 2, ai_LLM, is processed independently when language-specific, textual reasoning is critical, feeding CV_Adapt with contextual language inputs. Group 3, DTB, provides real-time environmental data in tandem with other components for decision-making real-world adaptation and simulation.

This integration strategy aligns with the concept of multimodal learning, which has been shown to enhance cognitive capabilities in artificial systems (Baltrušaitis et al., 2018). [103].

**2. Processing**

Parallel Computing: Distributes processing tensors (individually or in combination) across GPUs/TPUs, reflecting a modern approach to computational efficiency.

CV_Adapt serves as the central processor, functioning as a GPU, accepting combinations of tensors from other components based on context, and interpreting and adapting to input.

This architecture is consistent with the trend toward specialized hardware acceleration in AI systems (Jouppi et al., 2017). [104].

**3. Model Design:**

The combination of e_Generalization and e_Reasoning facilitates high-level abstraction and logic processing, guiding CV_Adapt to make decisions based on learned patterns and reasoning.

The ai_LLM is engaged at specific moments when linguistic or cognitive knowledge is required. The DTB enables environmental interaction and real-time simulation, allowing CV_Adapt to process and adapt dynamically. Dynamic Learning: CV_Adapt continually refines itself from input data streams, which aligns with the concept of continual learning in AI (Parisi et al., 2019). [105].

**4. AGI Output:**

Raw Data Output: After processing, CV_Adapt generates tensor data as output, serving as the 'DNA' for AGI processes, generated from a combination of reasoning, language understanding, generalization, and real-time simulation data.

This output represents a form of knowledge representation, a critical component of intelligent systems (Russell & Norvig, 2016). [106].

# Appendix 3

## DTB Model

| | |
|---|---|
| Neural Capacitance \| | Membrane's ability to store charge. |
| Membrane Potential Change \| | Voltage difference across the neuron's membrane. |
| Leak Conductance \| | Passive ion flow through always-open channels. |
| Equilibrium Potential \| | Voltage where ion flow reverses direction. |
| % Open Channels (Neuro.) \| | Proportion of neurotransmitter-gated channels open. |
| Driving Force (Current) \| | Difference between membrane and equilibrium potential. |
| Synapse Conductance \| | Ease of ion flow across a synapse. |
| Channel Closure Rate \| | Speed at which open ion channels return to closed. |
| Synaptic Weighting \| | Strength of synaptic connection/influence. |
| Pre-synaptic Spike Influence\| | Effect of pre-synaptic action potential on postsynaptic. |
| Time Since Past Spike \| | Interval between current time and prior action potential. |

Node Labels:

- A: Biological Neuron
- B: Data Acquisition
- C: Parameter Extraction
- D: Neural Cap.
- E: Mem. Pot.
- F: Leak Cond.
- G: Eq. Pot.
- H: % Open
- I: Drive Force
- J: Syn. Cond.
- K: Close Rate
- L: Syn. Weight
- M: Pre. Spike
- N: Time Spike
- O: Math Model
- P: Simulation
- Q: Digital Twin
- R: Vis/Analys
- S: Live Data

Diagram Structure:

1. A[Biological Neuron] --> 2. B(Data Acquisition):
A "Biological Neuron" box connects to a "Data Acquisition" box.

2. B(Data Acquisition) --> 3. C{Parameter Extraction}:
The "Data Acquisition" box connects to a "Parameter Extraction" diamond.

3. C{Parameter Extraction} --> 4. D[Neural Cap.]:
The "Parameter Extraction" diamond connects to a "Neural Cap." box.

3. C{Parameter Extraction} --> 4. E[Mem. Pot.]:
The "Parameter Extraction" diamond connects to a "Mem. Pot." box.

3. C{Parameter Extraction} --> 4. F[Leak Cond.]:
The "Parameter Extraction" diamond connects to a "Leak Cond." box.

3. C{Parameter Extraction} --> 4. G[Eq. Pot.]:
The "Parameter Extraction" diamond connects to a "Eq. Pot." box.

3. C{Parameter Extraction} --> 4. H[% Open]:
The "Parameter Extraction" diamond connects to a "% Open" box.

3. C{Parameter Extraction} --> 4. I[Drive Force]:
The "Parameter Extraction" diamond connects to a "Drive Force" box.

3. C{Parameter Extraction} --> 4. J[Syn. Cond.]:
The "Parameter Extraction" diamond connects to a "Syn. Cond." box.

3. C{Parameter Extraction} --> 4. K[Close Rate]:
The "Parameter Extraction" diamond connects to a "Close Rate" box.

1. C{Parameter Extraction} --> 4. L[Syn. Weight]:
The "Parameter Extraction" diamond connects to a "Syn. Weight" box.

3. C{Parameter Extraction} --> 4. M[Pre. Spike]:
The "Parameter Extraction" diamond connects to a "Pre. Spike" box.

3. C{Parameter Extraction} --> 4. N[Time Spike]:
The "Parameter Extraction" diamond connects to a "Time Spike" box.

4. D & E & F & G & H & I & J & K & L & M & N --> 5. O[Math Model]:
All the parameter boxes connect to a "Math Model" box.

5. O[Math Model] --> 6. P[Simulation]:
The "Math Model" box connects to a "Simulation" box.

6. P[Simulation] --> 7. Q[Digital Twin]:
The "Simulation" box connects to a "Digital Twin" box.

7. Q[Digital Twin] --> 8. R[Vis/Analys]:
The "Digital Twin" box connects to a "Vis/Analys" box.

8. B(Data Acquisition) --> 9. S[Live Data]:
The "Data Acquisition" box connects to a "Live Data" box.

9. S[Live Data] --> 7. Q[Digital Twin]:
The "Live Data" box connects to the "Digital Twin" box.

**Appendix 4**

## Gato (DeepMind)* vs. ai2agi (ai70000, Ltd.)**

| Feature | Gato (DeepMind) | ai2agi (ai70000, Ltd.) |
|---|---|---|
| **Type** | Single transformer model | Multi-system AI integration |
| **Architecture** | Sequence modeling transformer | Composite framework with specialized AI modules |
| **Inputs** | Text, images, actions (tokenized) | G_enhanced, R_enhanced, ai_LLM, DTB, CV_Adapt (***AGI dna*) |
| **Processing** | Autoregressive prediction | Functional mapping of subsystems |
| **Training Data** | 600+ multi-task datasets | Multi-modal cognitive, vision, and reasoning data |
| **Decision Making** | Implicit task representation | Dynamic weighting of subsystems |
| **Scope** | Generalist AI (many tasks, 1 model) | AGI-focused (reasoning, generalization, cognitive modeling) |
| **Biological Basis** | None | Digital Twin Brain (DTB) |
| **Vision System** | Pretrained CV embeddings | Adaptive cv (CV_Adapt) |
| **Goal** | Multi-task AI agent | AGI through modular intelligence |

** actual*
*** theoretical*

## Appendix 5

Ethical Implications:

- *Bias and Fairness* AGI systems, especially those incorporating ai_LLMs, may inherit and amplify biases present in training data. This could lead to unfair or discriminatory outcomes.
- *Autonomy and Control* AGI systems become more sophisticated, questions arise about their autonomy and the level of human control. Ensuring AGI remains aligned with human values is crucial.
- *Societal Impact* AGI could have profound societal impacts, including job displacement, economic disruption, and shifts in social structures.
- *Misuse and Malice* AGI could be misused for malicious purposes, such as developing autonomous weapons or creating sophisticated disinformation campaigns.
- *Transparency and Explainability* Understanding how AGI systems make decisions is essential for building trust and ensuring accountability.
- *DTB* raises ethical questions. What are the rights of a digital consciousness? How will digital consciousness effect the human condition?

## References


[1] Voss, P., and Jovanovic, M. (Aug 7th, 2023) Why We Don't Have AGI Yet, arXiv:2308.03598, https://doi.org/10.48550/arXiv.2308.03598

[2] Sukhobokov, A., et. al, (Jan 11th, 2024) A Universal Knowledge Model and Cognitive Architecture for Prototyping AGI, arxiv.org, https://www.arxiv.org/abs/2401.06256

[3] Cichocki, A., and, Kuleshov, A. P., (Aug 7th, 2020) Future Trends for Human-AI Collaboration: A Comprehensive Taxonomy of AI/AGI Using Multiple Intelligences and Learning Styles, arxiv:2008.04793[cs.AI], https://doi.org/10.48550/arXiv.2008.04793

[5] Srinvias, S.S. et., al., (Sep 17th, 2024) Sparks of Artificial General Intelligence (AGI) in Semiconductor Material Science: Early Explorations into the Next Frontier of Generative AI-Assisted Electron Micrograph Analysis, arxiv: 2409.12244v1[cs.CV], https://arxiv.org

[7] Latapie, H., and Kilic, O. (Sep 6th, 2020) A Metamodel and Framework for AGI, arxiv:2008.12879v2[cs.AI], https://www.arxiv.org

[8] Korzybski, A., Manhood of Humanity, The Science and Art of Human Engineering. E.P.Dutton & Company, NY (1921).

[9] Berlot-Atwell, I. (Apr 13th, 2021) Neuro-Symbolic VQA: A review from the perspective of AGI desiderata, arxiv:2104.06365v1[cs.LG], https://www.arxiv.org/2104.06365v1

[10] Hamilton, K., et. al., (Jan 1st, 2024) Is neuro-symbolic AI meetings its promises in natural language processing? A structured review. Semantic Web, 2024-01, Vol. 15(4), p.1265-1306, https://www.journals.sagepub.com/doi/full/10.3233/SW-223228

[12] Hagos, D.H., and Rawat, D.B., (Dec 1st, 2024) Neuro-Symbolic AI for Military Applications. IEEE transactions on artificial intelligence, 2024-12, Vol.5 (12), p. 6012-6026, https://www.ieeeexplore.ieee.org/document/10638797

[14] Alexander, S.A. (Oct 14th, 2020) The Archimedean trap: Why traditional reinforcement learning will probably not yield AGI, Sciendo.com, https://www.sciendo.com/article/10.2478/jagi-2020-0004

[16] Brown et. al. (May 28th, 2020) Language Models are Few-Shot Learners, arXiv.org, https://doi.org/10.48550/arXiv.2005.14165

[17] Stiennon, Nisan, et. al.(Feb 15th, 2022) Learning to summarize with human feedback. arXiv:2009.01325v3[cs.CL] https://doi.org/10.48550/arXiv.2009.0135v3

[19] Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T., Bishop, C., Carbune, V., and Abhinav, R., (Sep 1st, 2023) RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback, arXiv:2309.00267v1[cs.CL], https://doi.org/10.48550/arXiv.2309.00267

[20] Rusu et al. (Oct 22nd, 2022) Progressive Neural Networks, Semantics Scholar, https://www.semanticsscholar.org/paper/Progressive-Neural-Networks-Rusu-Rabinowitz/53c944e4e667170acc60ca1b31a0ec7151fe753

[22] Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I., (May 23rd, 2023) Improving F actuality and Reasoning in Language Models through Multi-agent Debate, GitHub.com, https://www.composable-models.github.io/lm_debate/

[23] Chollet, F., (2023) ArcPrize, https://www.arcprize.org

[24] Mumuni, A., and Mumuni, F., (Jan 6th, 2025) Large language models for artificial general intelligence (AGI): A survey of foundational principles and approaches. arXiv:2501.031151v1[cs.AI], https://www.arXiv.org

[26] Mathematics StackExchange, (2016) How was Zeno's paradox solved using the limits of infiniteseries?
https://www.math.stackexchange.com/questions/1623415/how-was-Zeno's-paradox-solved-using-the-limits-of- infinite-series

[27] Chen, J., Li, X., & Zhang, R. (2024). Efficient tensor processing in deep learning for computer vision applications. *Journal of Machine Learning Research*, 25(4), 301-318.

[28] Goodfellow, I., Bengio, Y., & Courville, A. (2023). *Deep Learning* (3rd ed.). MIT Press

[29] He, K., Zhang, X., Ren, S., & Sun, J. (2024). Deep residual learning for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3), 573-585.

[30] Zhou, X., Xu, L., & Song, Z. (2023). Vision transformers: A new era for deep learning. *IEEE Transactions on Image Processing*, 32(5), 1420-1434.

[31] Xie, et al. (Jun 14th, 2023). Towards AGI in Computer Vision: Lessons learned from GPT and Large Language Models, arXiv: 2306/08641cv1 [cs.CV]

[32] Atito, S., Awais, M., & Kittler, J. (2021). SiT: Self-supervised vision Transformer. arXiv preprint arXiv:2104.03602

[33] Paolo, G., Gonzalez-Billandon, J., & Kégl, B. (2024). Position: A Call for Embodied AI. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, F. Berkenkamp (Eds.), *Proceedings of the 41st International Conference on Machine Learning* (pp. 39493–39508). PMLR.

[34] Xi, J., He, Y., Yang, J., Dai, Y., & Chai, J. (2024). Teaching Embodied Reinforcement Learning Agents: Informativeness and Diversity of Language Use. arXiv preprint arXiv:2410.24218.

[35] Atanasov, A., & Parada, C. (2024, November 25). A Revolution in How Robots Learn. *The New Yorker*.

[36] Voss, P., and Jovanovic, M. (Aug 7th, 2023) Why We Don't Have AGI Yet, arXiv:2308.03598, https://doi.org/10.48550/arXiv.2308.03598

[37] Goertzel, B. (2007) Artificial general intelligence (Vol. 2, p.1) C. Pennachin (Ed.) New York: Springer.

[38] DeepMind. (2024, December 4). *Genie 2: A large-scale foundation world model*. Retrieved from https://deepmind.com/discover/blog/genie-2-a-large-scale-foundation-world-model/

[39] Shang et al. (Jan 14th, 2025) Biologically-Inspired Technologies: Integrating Brain-Computer Interface and Neuromorphic Computing for Human Digital Twins, arXiv: 2410.23639v2 [cs.HC]

[40] Li et al. (Feb 20th, 2025) How Far are LLMs from Being Our Digital Twin? A Benchmark for Persona-Based Behavior Chain Simulation, axXiv: 2502.14642v1 [cs.CL]

[41] Hakim et al. (Jan 15th, 2025) ANSR-DT: An Adaptive Neuro-Symbolic Learning and Reasoning Framework for Digital Twins, axXiv: 2501.08561v1[cs.AI]

[42] Qi et al. (2024) Optimal signal transmission and timescale diversity in a model of human brain operating near criticality. arXiv preprint arXiv:2412.17043.

[43] Xie et al. (2023) Towards AGI in Computer Vision: Lessons Learned from GPT and Large Language Models. arXiv preprint arXiv:2306.08641

[44] Liu, Z. (2023). Towards building artificial general intelligence via multimodal deep learning. arXiv preprint arXiv:2305.03713.

[45] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

[46] AI21 Labs. (2024). Introducing Jamba: AI21 Labs' groundbreaking production-grade long context model. Retrieved from https://www.ai21.com/blog/introducing-jamba-ai21-labs-groundbreaking-production-grade-long-context-model

[45] Berlot-Attwell, I. (2021). Neuro-Symbolic VQA: A review from the perspective of AGI desiderata. arXiv preprint arXiv:2104.06365.

[47] Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of Physiology, 117(4), 500–544.

[49] Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. MIT.

[50] Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). Neuronal dynamics: From single neurons to networks and models of cognition. Cambridge University Press.

[51] Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. The Journal of Neuroscience, *18*(24), 10464–10472. )

[52] Malenka, R. C., & Bear, M. F. (2004). LTP and LTD: An embarrassment of riches. *Neuron, 44*(1), 5–21.

[53] Sporns, O. (2011). *Networks of the Brain*. MIT Press.

[54] Bullmore, E., & Sporns, O. (2009). Complex brain networks: Graph theoretical analysis of structural and functional systems. Nature Reviews Neuroscience, 10(3), 186–198.

[55] Fields, R. D., & Stevens-Graham, B. (2002). New insights into neuron-glia communication. *Science, 298*(5593), 556–562.

[56] Haydon, P. G. (2001). GLIA: Listening and talking to the synapse. *Nature Reviews Neuroscience, 2*(3), 185–193.

[57] Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., LaMantia, A. S., McNamara, J. O., & White, L. E. (2018). *Neuroscience* (7th ed.). Sinauer Associates.

[58] Siegel, G. J., Agranoff, B. W., Albers, R. W., Fisher, S. K., & Uhler, M. D. (Eds.). (1999). Basic neurochemistry: Molecular, cellular and medical aspects (6th ed.). Lippincott-Raven.

[59] Carnevale, N. T., & Hines, M. L. (2006). *The NEURON Book*. Cambridge University Press.

[60] Goodman, D., & Brette, R. (2009). The Brian simulator. *Frontiers in Neuroscience, 3*, 26.

[61] Gewaltig, M. O., & Diesmann, M. (2007). NEST (NEural Simulation Tool). *Scholarpedia, 2*(4), 1430. (An article about the NEST simulator)

[62] Dehaene, S. (2014). *Consciousness and the Brain: Deciphering How the Brain Codes Our Thoughts*. Viking.

[63] Baars, B. J. (1988). *A cognitive theory of consciousness*. Cambridge University Press.

[64] O'Reilly, R. C., & Munakata, Y. (2000). Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain. MIT Press.

[65] Goodman, D. F. M., & Brette, R. (2009). The Brian simulator. *Frontiers in Neuroscience, 3*, 26. https://doi.org/10.3389/neuro.01.026.2009

[66] Haydon, P. G. (2001). GLIA: Listening and talking to the synapse. *Nature Reviews Neuroscience, 2*(3), 185-193. https://doi.org/10.1038/35058528

[67] Gewaltig, M.-O., & Diesmann, M. (2007). NEST (Neural Simulation Tool). *Scholarpedia, 2*(4), 1430.

[68] Sporns, O. (2010). *Networks of the Brain*. The MIT Press.

[69] Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., LaMantia, A.-S., McNamara, J. O., & Williams, S. M. (Eds.). (2004). *Neuroscience* (3rd ed.). Sinauer Associates.

[70] Alan Turing (1950). *Computing Machinery and Intelligence*

[71] Warren McCulloch & Walter Pitts (1943). *A Logical Calculus of the Ideas Immanent in Nervous Activity*

[72] Stuart Russell & Peter Norvig (1995). *Artificial Intelligence: A Modern Approach*

[75] Berber, J. (2025). *Founding of Safe Superintelligence (SSI)*. *The Wall Street Journal*. Retrieved from https://www.wsj.com/tech/ai/ai-safe-superintelligence-startup-ilya-sutskever-openai-2335259b

[76] Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Gur-Ari, H., Sadeghi, I., ... & de Freitas, N. (2022). A generalist agent.arXiv preprint arXiv:2205.06175.

[77] Berners-Lee, T., & Fischetti, M. (2000). Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor. HarperCollins Publishers.

[78] McCarthy, J. (1971). GENERALITY IN ARTIFICIAL INTELLIGENCE. Stanford University. Retrieved from https://www-formal.stanford.edu/jmc/generality/generality.html

[79] Altman, S. (2023, October 26). The future of AI. *Sam Altman*. https://blog.samaltman.com/the-future-of-ai

[80] Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3(3), 210-229

[81] Friston, K. J., Harrison, L., & Penny, W. (2003). Dynamic causal modelling. *NeuroImage*, *19*(4), 1273-1302.

[82] Sporns, O. (2018). *Network neuroscience*. MIT press.

[83] Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (2013). *Principles of neural science* (5th ed.). McGraw-Hill Education.

[84] Allen, N. J., & Barres, B. A. (2009). Neuroscience: Glia—more than just brain glue. *Nature*, *457*(7230), 675-677.

[85] Hubel, D. H. (1988). *Eye, brain, and vision*. Scientific American library.

[86] Posner, M. I. (1980). Orienting of attention. *Quarterly journal of experimental psychology*, *32*(1), 3-25.

[87] LeDoux, J. E. (2012). Rethinking the emotional brain. *Neuron*, *73*(4), 650-676.

[88] Neuromodulation: Nieuwenhuis, S., Aston-Jones, G., & Cohen, J. D. (2005). Decision making, the P3, and the locus coeruleus norepinephrine system. Psychological bulletin, 131(4), 1 510.

[89] Carnevale, N. T., & Hines, M. L. (2006). *The NEURON book*. Cambridge University Press.

[90] Goodman, D. F., & Brette, R. (2009). The Brian simulator. *Frontiers in Neuroscience*, *3*, 192.

[91] Izhikevich, E. M., Edelman, G. M., & Krichmar, J. L. (2004). Large-scale model of the human thalamocortical system. *Proceedings of the National Academy of Sciences*, *101*(3), 714–719.

[92] Destexhe, A., Rudolph, M., & Paré, D. (2003). The high-conductance state of cortical neurons. *Nature Reviews Neuroscience*, *4*(10), 739–751.

[94] Marder, E. (2012). Neuromodulation of neuronal circuits: back to the future. *Neuron*, *76*(1), 1–11.

[95] Tononi, G. (2008). Consciousness as integrated information: a provisional manifesto. *Biological Bulletin*, *215*(3), 216–242.

[96] Miller, E. K., & Cohen, J. D. (2001). An integrative theory of prefrontal cortex function. Annual Review of Neuroscience, 24(1), 167–202.

[97] Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., ... & Destexhe, A. (2007). Simulation of networks of spiking neurons: a review of tools and strategies. Journal of Computational Neuroscience, 23(3), 349–398.

[98] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, J., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Andreev, G., Fort, Q., Weissenborn, J., Houlsby, N., Steiner, A., & al., et. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[99] Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for modern deep learning research. *arXiv preprint aXiv:1904.09714*.

[100] Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: Computational and mathematical modeling of neural systems*. MIT Press.

[101] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[102] Marcus, G. (2020). Next generation artificial intelligence and the future of AGI. *arXiv preprint arXiv:2002.06277*

[103] Baltrušaitis, T., Ahuja, C., & Morency, L. P. (2018). Multimodal machine learning: A survey and taxonomy. IEEE transactions on pattern analysis and machine intelligence, 1 41(2), 424-443. 2

[104] Jouppi, N. P., Young, C., Patil, D., Patterson, D., Agrawal, G., Bajwa, R., ... & Williams, C. (2017). In-datacenter performance analysis of a tensor processing unit™. In *2017 ISCA* (pp. 1-12).

[105] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning of artificial agents: A review. *Neural networks*, *113*, 54-71.

[106] Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson Education.

[107] Nadim, F., & Bucher, D. (2014). Neuromodulation of neurons and synapses. *Current Opinion in Neurobiology*, *29*, 48–56.

[108] Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., & Le, Q. V. (2012). Large scale distributed deep networks. In *Advances in neural information processing systems* (Vol. 25).

[109] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444

[110] Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for modern deep learning research. *arXiv preprint arXiv:1904.09714*

[111] OpenAI (2025, Jan 21st), Announcing the Stargate Project, Retrieved from: https://openai.com/index/announcing-the-stargate-project/

[112] Zhu, H. Y., Hieu, N. Q., Hoang, D. T., Nguyen, D. N., & Lin, C. T. (2023). A Human-Centric Metaverse Enabled by Brain-Computer Interface: A Survey. IEEE Communications Surveys & Tutorials, Vol, 00, No. 00.

[113] Xiong, H., Chu, C., Fan, L., Song, M., Zhang, J., Ma, Y., Zheng, R., Zhang, J., Yang, Z., & Jiang, T. (2023). Digital twin brain: a bridge between biological intelligence and artificial intelligence. arXiv preprint arXiv:2308.01241.

[114] Fekonja, L. S., Schenk, R., Schröder, E., Tomšič, S., & Picht, T. (2024). The digital twin in neuroscience: From theory to tailored therapy. Frontiers in Neuroscience, 18. https://doi.org/10.3389/fnins.2024.1454856

[115] Beniaguev, D., Segev, I., & London, M. (2021). Single cortical neurons as deep artificial neural networks.*Neuron*, 109(17), 2727-2739.e3.

[116] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2016). Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289.

[117} Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on knowledge and data engineering, 22(10), 1345-1359.

[118] Rahman, A., Mahir, S. H., Tashrift, M. T. A., Aishi, A. A., Karim, M. A., Kundu, D., ... & Eidmum, M. Z. A. (2025). Comparative Analysis Based on DeepSeek, ChatGPT, and Google Gemini: Features, Techniques, Performance, Future Prospects. arXiv preprint arXiv:2503.04783.

[119] Sutton, R. S., Bowling, M., & Pilarski, P. M. (2023). The Alberta plan for AI research. *arXiv preprint arXiv:2208.11173*.

[120] NVIDIA Corp. (2025, March 10). NVIDIA Corp (NVDA) stock price, news, quote & history - Yahoo Finance. Yahoo Finance. https://finance.yahoo.com/quote/NVDA/

[121] Markram, H., Wang, Y., & Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. Proceedings of the National Academy of Sciences, 95(9), 5323-5328.

[122] Koch, C., & Segev, I. (Eds.). (2000). Methods in neuronal modeling: From ions to networks (2nd ed.). MIT press.

[123] Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? IEEE Transactions on Neural Networks, 15(5), 1063-1070.

[125] Allen, N. J., & Eroglu, C. (2017). Cell biology of astrocyte-synapse interactions. Neuron, 96(3), 697–708. https://doi.org/10.1016/j.neuron.2017.09.056

[126] Wilson, H. R., & Cowan, J. D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. Biophysical journa, 12(1), 1-24.

[127] Buzsáki, G., & Draguhn, A. (2004). Neuronal oscillations in cortical networks. *Science*, *304*(5679), 1926-1929.

[128] Breakspear, M. (2017). Dynamic models of large-scale brain activity. Nature neuroscience, 20(3), 340-352.

[129] Tsodyks, M., Pawelzik K., Markram H., ; Neural Networks with Dynamic Synapses. *Neural Compute* 1998; 10 (4): 821–835. doi: https://doi.org/10.1162/089976698300017502

[130] Araque, Alfonso et al.Tripartite synapses: glia, the unacknowledged partnerTrends in Neurosciences, Volume 22, Issue 5, 208 - 215

[131] Destexhe, A., Rudolph, M. & Paré, D. The high-conductance state of neocortical neurons *in vivo*. *Nat Rev Neurosci* 4, 739–751 (2003). https://doi.org/10.1038/nrn1198

[132] Fetty, N., (2024, Apr 8th) Yann LeCun emphasizes the promise of AI. https://www.nyas.org/ideas-insights/blog/yann-lecun-emphasizes-the-promise-ai/

**W. McCulloch & W. Pitts** (1943). *A Logical Calculus of the Ideas Immanent in Nervous Activity*:

"We suggest that any system of ideas, including the nervous system, can be represented by a set of simple logical operations." (McCulloch & Pitts, 1943, p. 120)

**Alan Turing** (1950). *Computing Machinery and Intelligence*:

"I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted." (Turing, 1950, p. 457)

**Claude Shannon** (1950). *Programming a Computer for Playing Chess*:

"The idea of programming a computer to play chess is an important part of the development of machines that can simulate human thought." (Shannon, 1950, p. 258)

**John McCarthy** (1959). *Programs with Common Sense*:

Artificial intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs." (McCarthy, 1959, p. 78)

**Marvin Minsky** (1969). *Perceptrons: An Introduction to Computational Geometry*:

"The perceptron is a simple neural network that is capable of learning to recognize patterns by adjusting its weights based on the errors in its output." (Minsky, 1969, p. 15)

**Terry Winograd** (1972). *Understanding Natural Language*:

"The goal is to make a machine that can use human language in a way that reflects human understanding." (Winograd, 1972, p. 10)

**Arthur Samuel** (1959). *Some Studies in Machine Learning Using the Game of Checkers*:

"The ability of a machine to improve its performance through experience and feedback is one of the hallmarks of intelligent behavior." (Samuel, 1959, p. 220)

**Allen Newell & Herbert A. Simon** (1976). *Computer Science as Empirical Inquiry: Symbols and Search*:

"The physical symbol system hypothesis is that a physical symbol system has the necessary and sufficient means for general intelligent action." (Newell & Simon, 1976, p. 118)

**Stuart Russell & Peter Norvig** (1995). *Artificial Intelligence: A Modern Approach*:

"Artificial intelligence is the study of agents that receive precepts from the environment and take actions to maximize their chances of success." (Russell & Norvig, 1995, p. 1)

**Ben Goertzel** (2007). *SingularityNET:*

"Artificial intelligence is not just a field of study, it is the pursuit of creating machines that can think, learn, and adapt to new situations, just as human beings do." (Goertzel, 2001, p. 5)

**Sam Altman** (2023) *OpenAI*:

“Our goal is to ensure that AGI, when it is created, benefits all of humanity. We want to build AI systems that are aligned with human values and can be trusted to act in ways that are beneficial to society." (Altman, 2023.)

**Elon Musk** (2023) *xAI*:

"We need to regulate AGI development carefully, because once we create something that is smarter than the best human minds in every field, it will be too late to fix the problems if it goes wrong." (Musk, 2023.)

**Ilya Sutskever** (2023) (*OpenAI / Founder of Safe Superintelligence*):

"As we approach AGI, the most important question is not whether it will be capable of performing tasks at human-level intelligence, but whether we can ensure its alignment with human goals, making sure it acts in a way that is safe and beneficial." (Sutskever, 2023)

**Demis Hassabis** (2024) *DeepMind*:

"The path to AGI is not just about scale or more data; it's about understanding the fundamental principles of how intelligence works, which could allow us to create systems that possess general reasoning capabilities." (Hassabis, 2024)