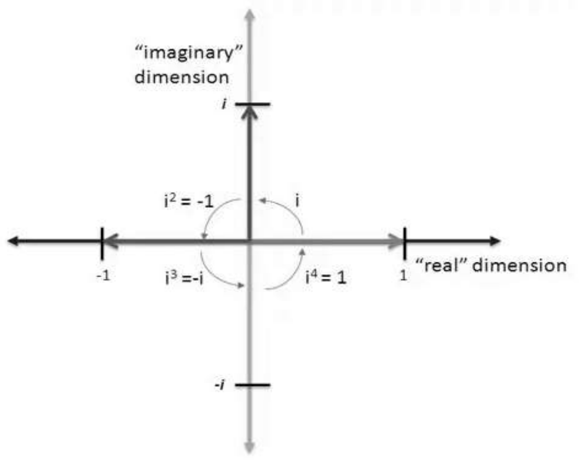
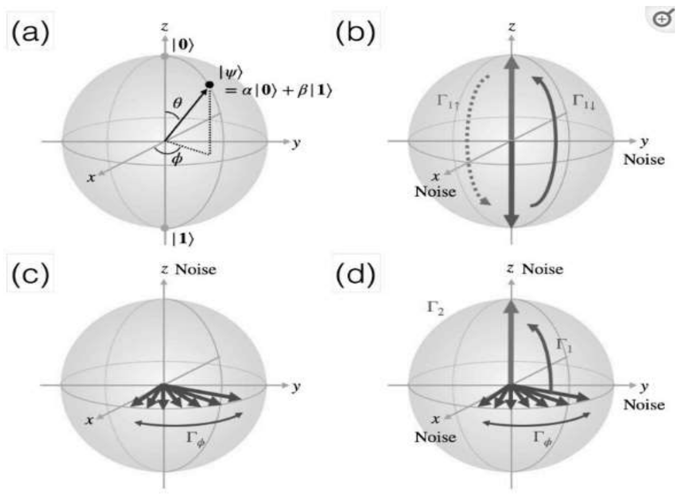
Quaternion data compression, The math.

Adding entropy to a system seems to be the most optimal way to create various patterns in large language model transformers. This proves useful but creates inherent problems in systems that require patterns based on high quality, real time inputs. Here an alternate approach is proposed.

Quaternions are inherently multi-dimensional objects, they cannot be treated like linear data. This brings a disconnect between visualizing high contextual data, as the linear systems must strip all contextual information and then fine tune their prediction algorithms to create context and add it leading to many steps in processing and computation.

Trying to visualize quaternion data from that perspective is limiting, Lets try a different approach. Imagine each token not as a single string of text, but a marble of data. This adds a whole additional dimension available for data.





Let's now break down how we do this, step by step. The Theory paper will then explain how this data will be transformed into a 2D packet, while still retaining its valuable multi dimensional data.

A Mathematical Proof on the Compression of 2D Data Arrays into Quaternions

Objective: To prove that converting a 2D real-valued data array into a quaternion representation is a form of computational compression, where the effective data size is reduced according to the square-root time-space relationship.

1. Foundational Concepts and Definitions

We will draw from the following sources:

1. **Time-Space Simulation** from *Simulating Time With Square-Root Space* (Williams, 2025).
2. **Quaternion Representation & Convolution** from *Deep Quaternion Networks* (Gaudet & Maida, 2018).
3. **Observer-Dependent Contextuality** from *A quantum semantic framework for natural language processing* (Agostino et al., 2025).
4. **Structured Data Packing** as an analogy from *Circle packing in regular polygons* (Amore, 2022).

Definitions:

The Core Idea: Time vs. Space

In computer science, there's a fundamental trade-off between time (how many steps an algorithm takes) and space (how much memory it needs). For decades, a key question has been: if you have a very long computation, can you do it with a surprisingly small amount of memory?

The Theoretical Limit - A New Law of Computation

Before we can build our framework, we must understand the fundamental "law of physics" from theoretical computer science that makes it possible. This law governs the relationship between the *time* a task takes and the *memory* it requires.

Time and Space: The Two Resources of Computing

Every computational task consumes two basic resources:

- **Time:** How many steps an algorithm must execute to finish. For a data array, this is proportional to the number of elements it has to process.
- **Space:** How much memory (RAM, storage) the algorithm needs to hold information while it's working.

Imagine baking a cake. The **time** is the total minutes from mixing to frosting. The **space** is the amount of counter space you need for your bowls, ingredients, and utensils. For decades, it was assumed that if you wanted to cut down on time, you'd probably need more space, and vice-versa.

The Old Rule vs. The New Rule

For a long time, the best-known relationship for a generic computation was that a task taking steps could be done with memory proportional to $t/logt$. This was a good improvement, but not revolutionary.

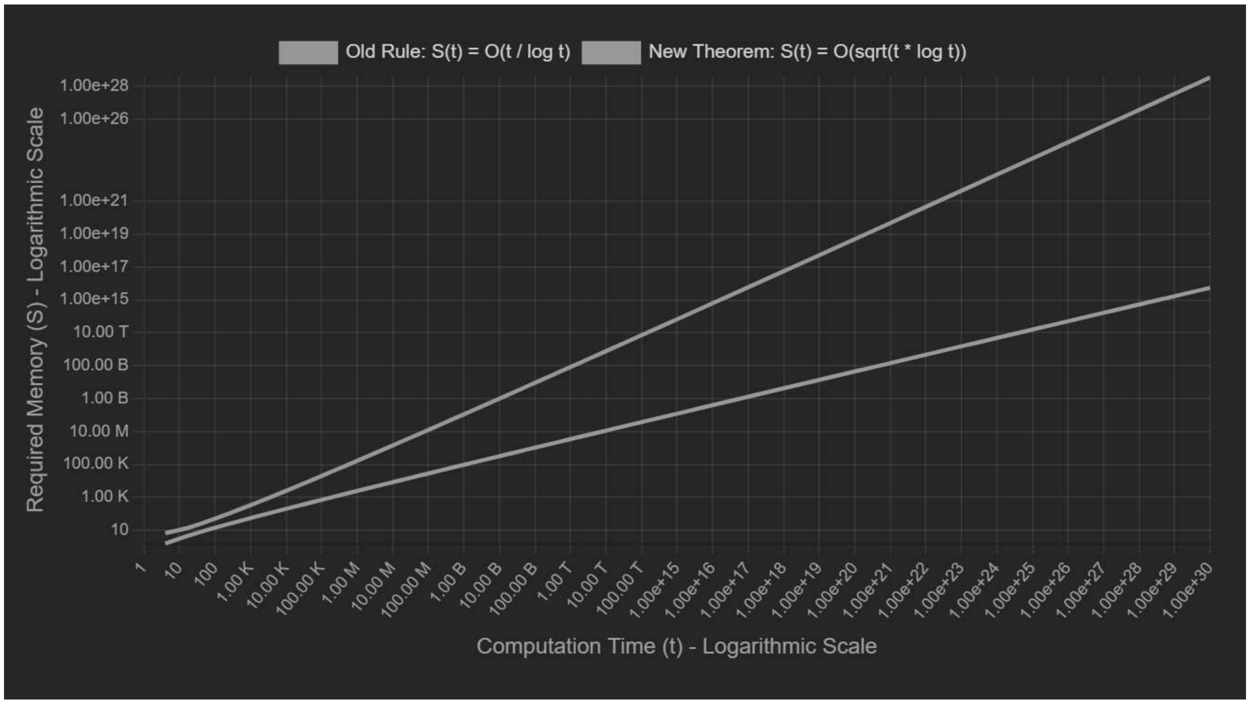
However, the paper *Simulating Time With Square-Root Space* (Williams, 2025) introduced a new, much more powerful rule.

- **The Square-Root Compression Theorem:** Any computation that runs in time t can be simulated using an amount of memory ($space$), S , defined by the relationship:

  $$S(t) = O(\sqrt{t \, log \, t})$$

This is a seismic shift. Let's use a concrete example. Suppose you have a computation that takes **1 trillion steps** ($t = 10^{12}$).

- The old rule suggested you'd need space on the order of $10^{12}/log(10^{12})$, which is still a massive number.
- The new theorem guarantees it can be done with space proportional to $\sqrt{10^{12}}$, which is only **1 million**



This means that for any long process, an incredibly more efficient computational pathway *must exist*. It's a theoretical guarantee. Our job now is to find a practical way to build it.

The Practical Tool - Spheres (Quaternions) packing

What is the most efficient way to pack identical, non-overlapping circles into a confined shape?

This isn't just a mathematical curiosity. It has direct applications in:

- **Physics:** Modeling how granular materials like sand or colloids behave.
- **Biology:** Understanding the arrangement of cells or pores on pollen grains (the original "Tammes problem").
- **Logistics:** Optimizing the storage of cylindrical objects in a container.

The Goal: Maximizing Density

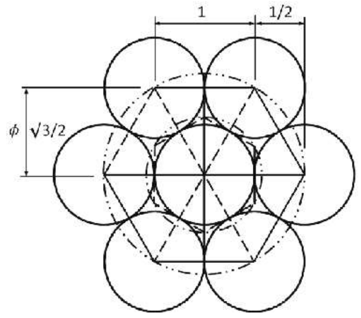
The primary goal is to maximize the **packing fraction (ρ).** This is the ratio of the total area covered by the circles to the total area of the container.

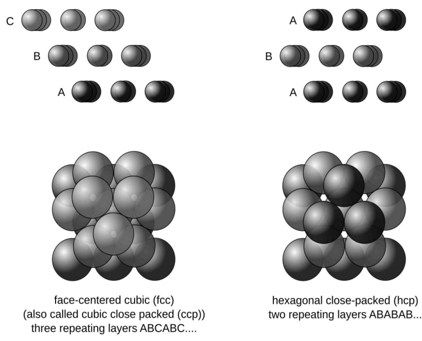
$\rho \equiv \frac{A(r,\sigma)}{N\pi r2}$

Where:

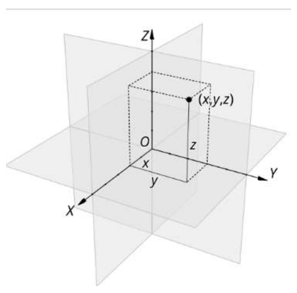
- N is the number of circles.
- r is the radius of each circle.
- A(r, σ) is the total area of the regular polygon container with σ sides.

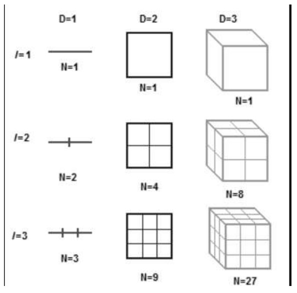
A packing fraction of $\rho = 1.0$ would mean the circles cover 100% of the area, which is impossible due to the gaps between them.



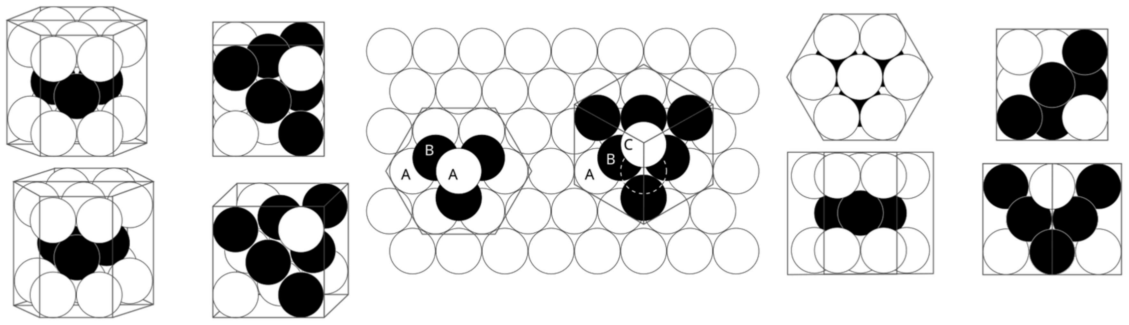

C
B
A
A
B
A
face-centered cubic (fcc)
(also called cubic close packed (ccp))
three repeating layers ABCABC....
hexagonal close-packed (hcp)
two repeating layers ABABAB...

The **Kepler conjecture** is a mathematical theorem about sphere packing in three-dimensional Euclidean space.





It states that no arrangement of equally sized spheres filling space has a greater average density than that of the cubic close packing (face-centered cubic) and hexagonal close packing arrangements. The density of these arrangements is around 74.05%.


B
A
A
C
A
B