Prospects of LLMs in Fundamental Physics

Siavash Golkar & Mariel Pettee

February 22nd, 2024

BERKELEY LAB
Bringing Science Solutions to the World

FLATIRON INSTITUTE

Unanswered questions

Unanswered questions

What is the nature of dark matter/energy?

Why is the universe so matter-dominated?

Why is gravity so weak?

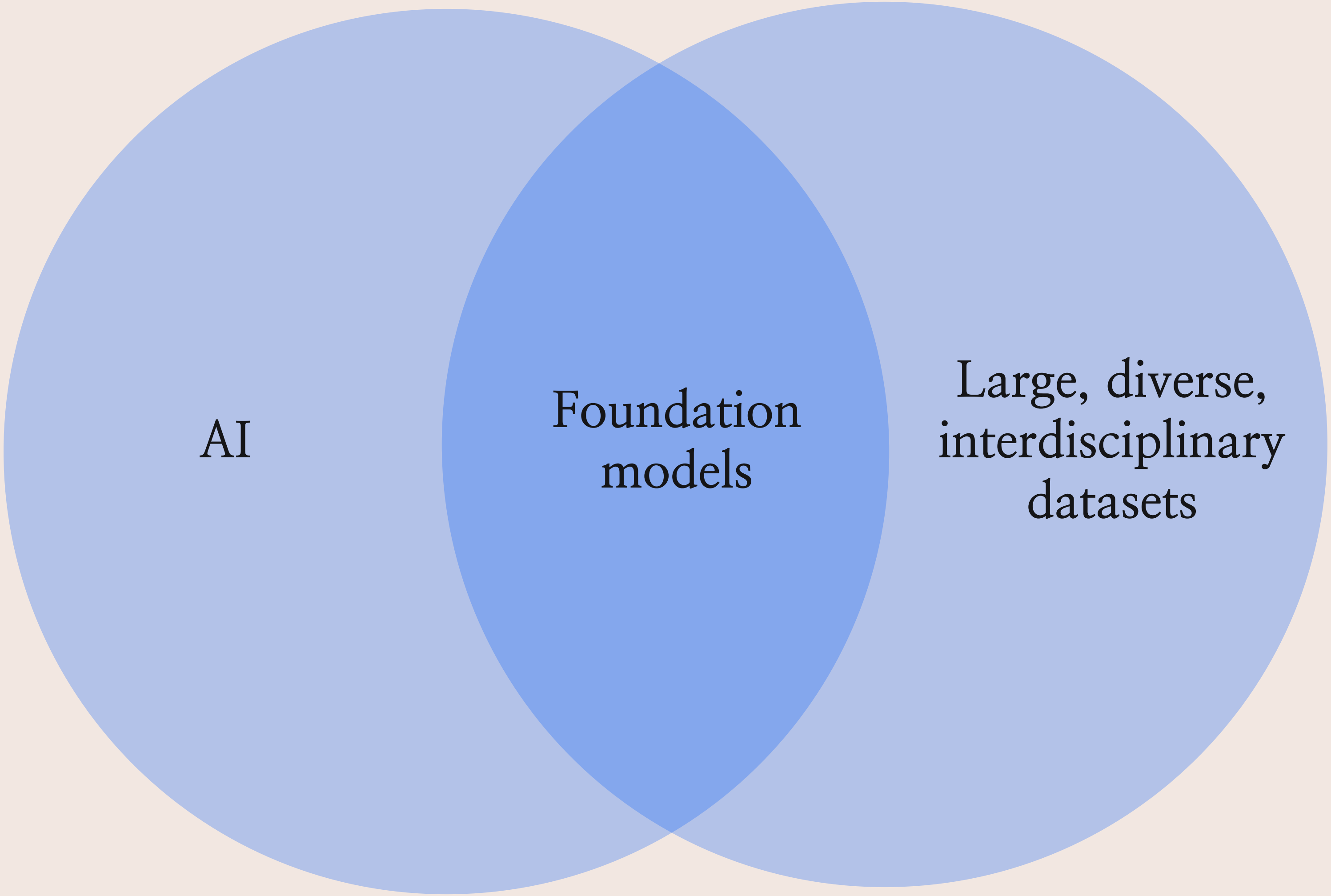
Why is the Higgs mass so light?

How does gravity operate in the quantum regime?

What's up with neutrino masses?

We don’t know yet how to answer these questions…

but we have **a ton of interesting data** to comb through, and it looks like **AI** will be essential to that process.


AI
Foundation models
Large, diverse, interdisciplinary datasets

Foundation models are pre-trained on large and diverse datasets.

Foundation models are (quasi-)generalists.

They can **make good predictions out-of-the-box** on several unseen tasks, and they can be **fine-tuned** for better performance on a variety of other tasks.

Why might we want to build a foundation model for fundamental physics research?

1. Efficiency:

A foundation model could provide a **better starting point than training from scratch** for training new AI models on physics data, particularly if training data is limited or expensive.

2. Interdisciplinarity:

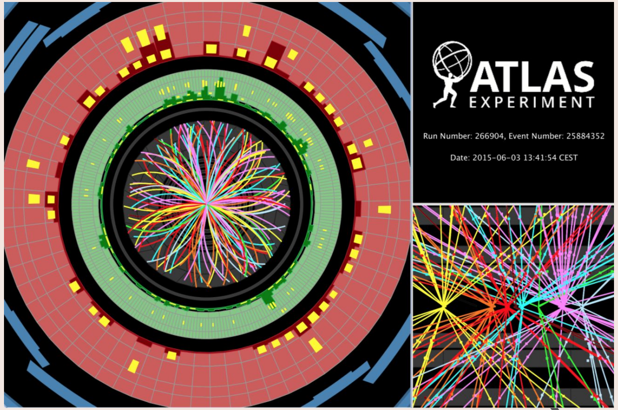
A foundation model could **reveal connections in data** collected from disparate instruments or even sub-disciplines, **transcending the scope of any one particular experiment's vantage point.**

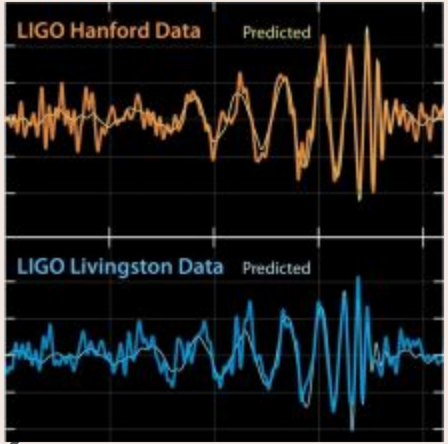
3. Creativity:

A foundation model could **foster more interactive, playful, curiosity-driven experiences with our data**, challenging our default assumptions about how typical analyses should be done and **inviting new, highly creative strategies.**

It’s (relatively) straightforward to imagine constructing a foundation model trained on data that can be neatly combined via a standard input format (e.g. text, 2D images).

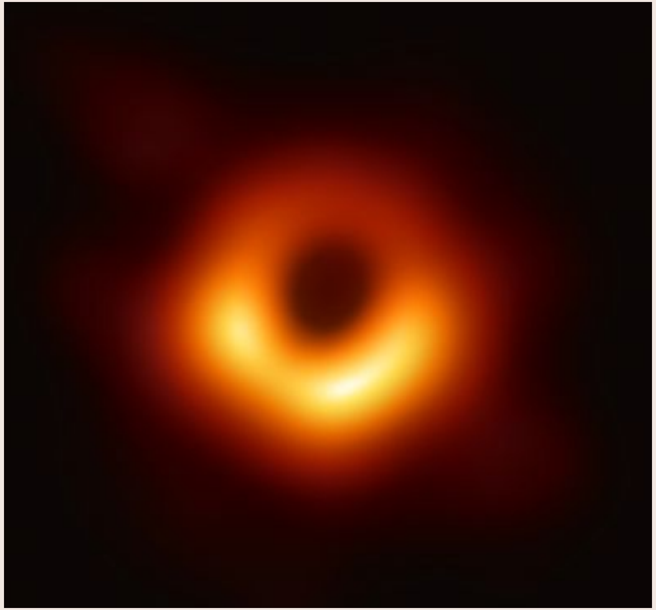
But physics data is far more heterogeneous and multi-scale.

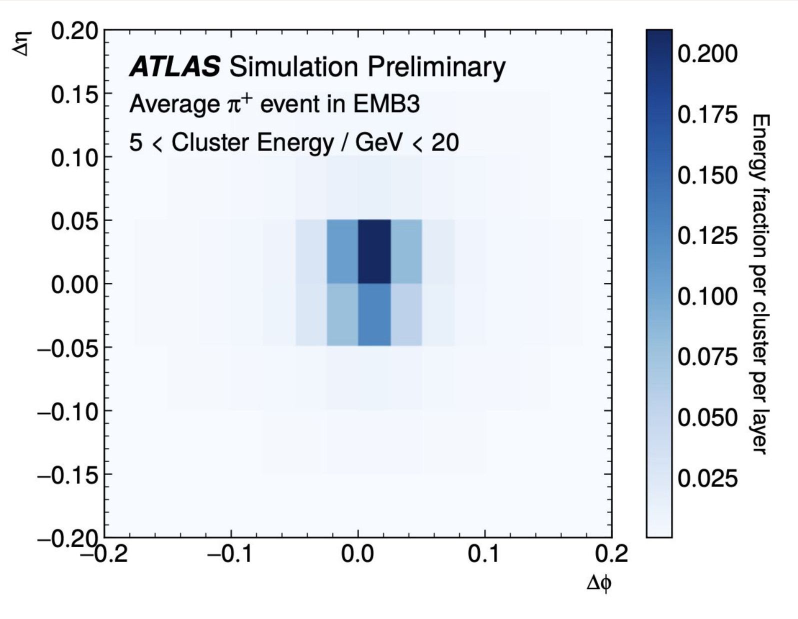

ATLAS
EXPERIMENT
Run Number: 266904, Event Number: 25884352
Date: 2015-06-03 13:41:54 CEST


LIGO Hanford Data
Predicted
LIGO Livingston Data
Predicted

Science GPT???

Fundamental physics data takes many forms and requires context.

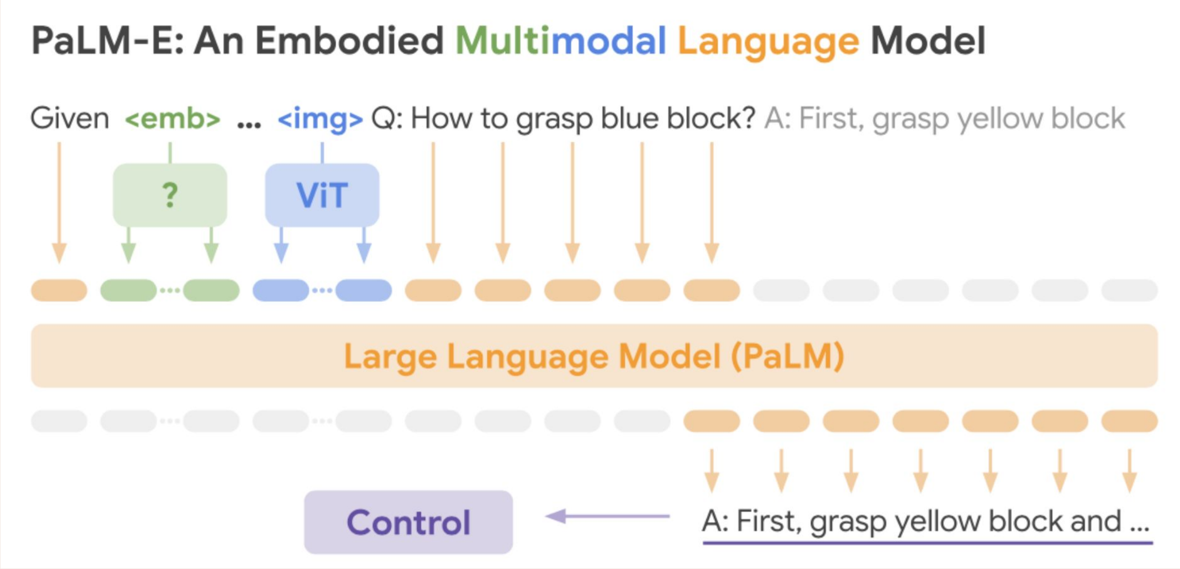



ATLAS Simulation Preliminary
Average π^+ event in EMB3
5 < Cluster Energy / GeV < 20
Δη
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.2
−0.1
0.0
0.1
0.2
Δϕ
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
Energy fraction per cluster per layer

We need to find a way to combine all of our inputs into a **single, shared "embedding space"** – i.e. a common language that our foundation model understands.

We need to find a way to combine all of our inputs into a **single, shared "embedding space"** – i.e. a common language that our foundation model understands.

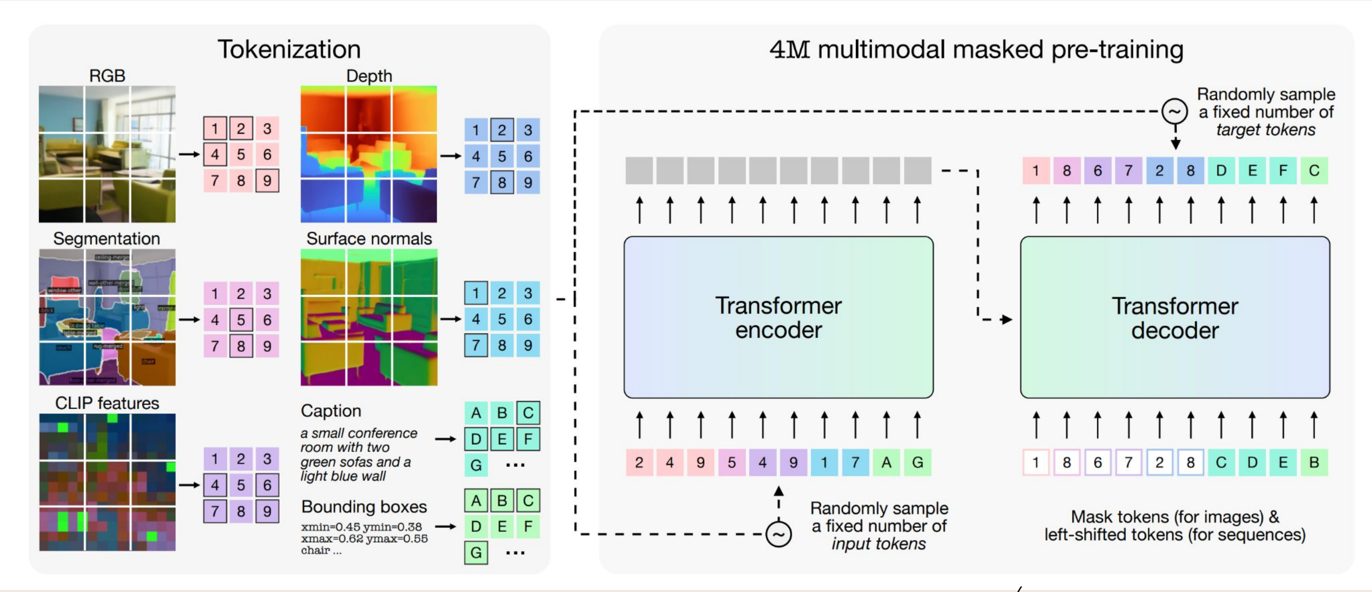
For a full-scale ScienceGPT, this will likely involve some specialized embedding structures, e.g.:



(Driess et. al., PaLM-E, 2023)

We need to find a way to combine all of our inputs into a **single, shared "embedding space"** – i.e. a common language that our foundation model understands.

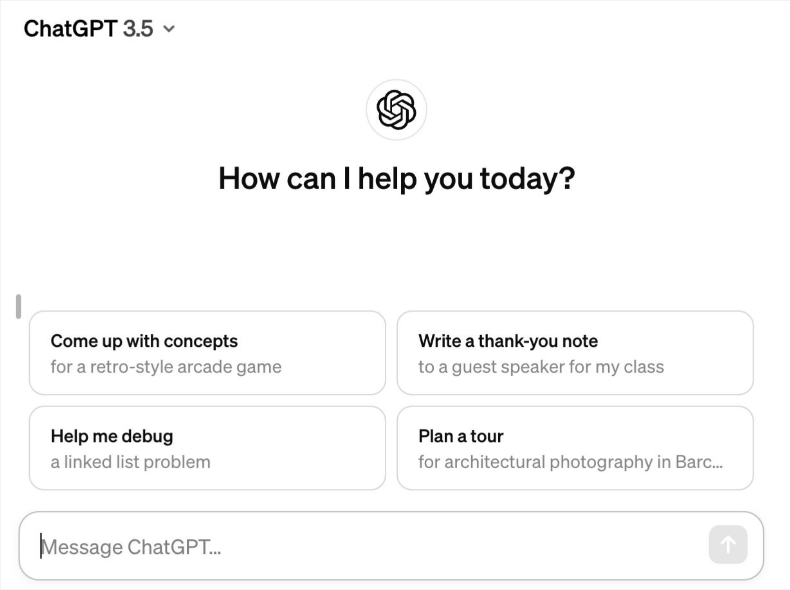
For a full-scale ScienceGPT, this will likely involve some specialized embedding structures, e.g.:



(Mizrahi & Bachmann et. al., [4M](), 2023)

We need to find a way to combine all of our inputs into a **single, shared "embedding space"** – i.e. a common language that our foundation model understands.

But for today, let's imagine we do the simplest version of this: **treat all your data as text.**



MA **You**

What is 3582 × 2911?

Wrong! It's 10,427,202.

ChatGPT

3582 multiplied by 2911 equals 10,425,302.

If we repeat this many times, we'll find that it will only give the correct answer around **4% of the time.**

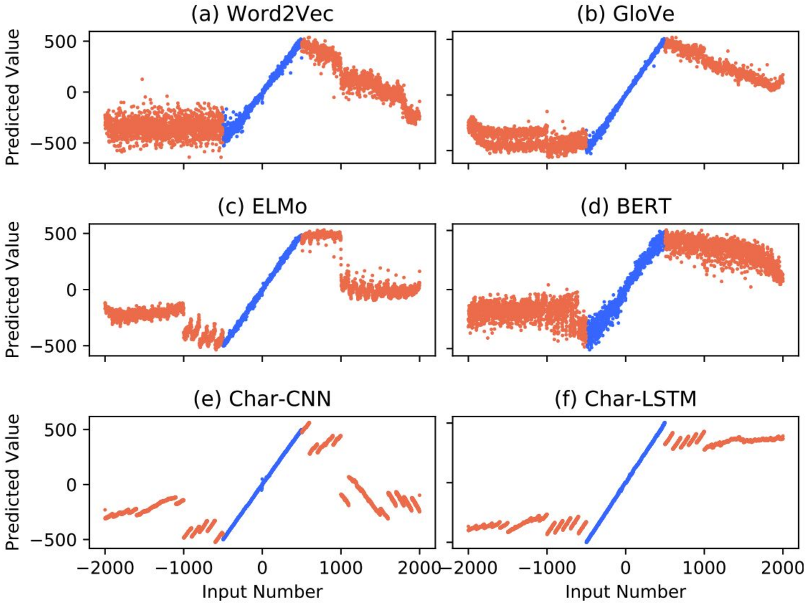
Faith and Fate: Limits of Transformers on Compositionality. arXiv:2305.18654 [cs.CL] (2023)

Large Language Models (LLMs) struggle to understand what makes numbers different from other kinds of text.

Existing embeddings can't generalize out-of-distribution…

Fixed Word Embeddings

Contextual Word Embeddings

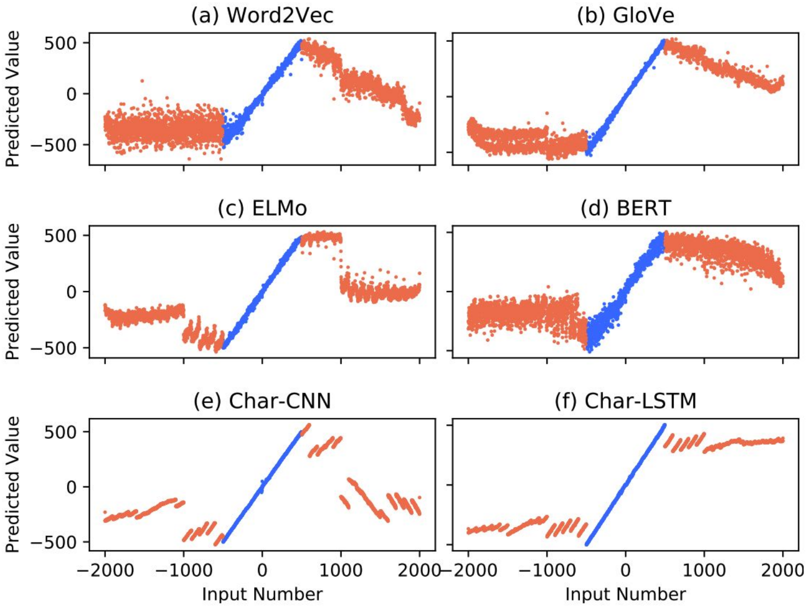


[1] Do NLPs Know Numbers? Probing Numeracy in Embeddings. https://aclanthology.org/D19-1534.pdf
[2] NumGPT: Improving Numeracy Ability of Generative Pre-trained Models. arXiv:2109.03137 [cs.CL].

Existing embeddings can't generalize out-of-distribution... ...and they behave erratically.

Fixed Word Embeddings

Contextual Word Embeddings



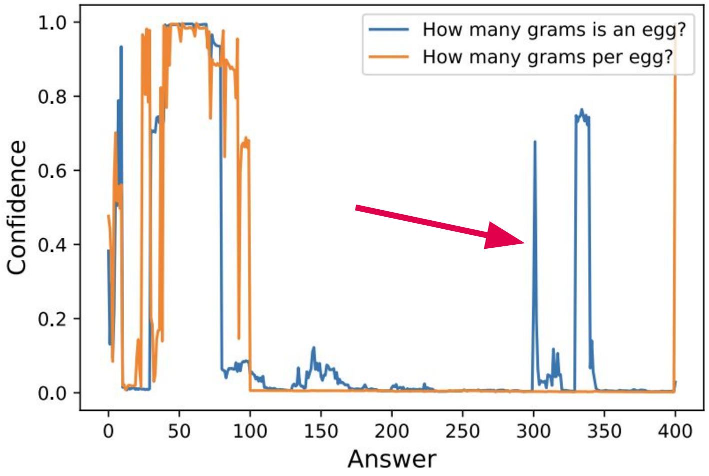


Figure 1: The confidence for GPT answering the questions related to the weight of an egg. The fluctuated curve reflects that GPT does not capture the continuous property of numbers.

[1] Do NLPs Know Numbers? Probing Numeracy in Embeddings. https://aclanthology.org/D19-1534.pdf
[2] NumGPT: Improving Numeracy Ability of Generative Pre-trained Models. arXiv:2109.03137 [cs.CL].

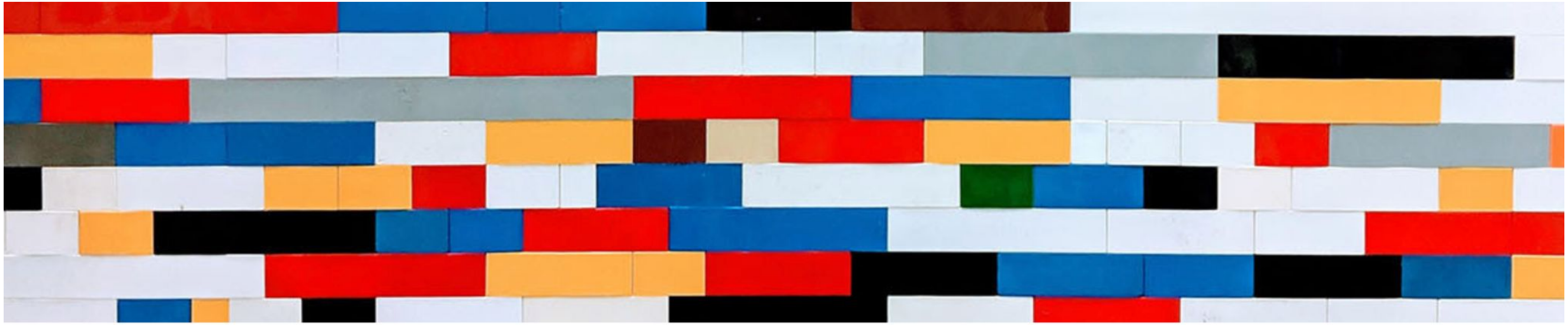
One core problem is the need to map every number onto a finite set of “tokens”.

Dedicated numerical encodings see trade-offs between accuracy, range, and vocabulary size.

Encoding	3.14	$-6.02.10^{23}$	Tokens / coefficient	Size of vocabulary
P10	`[+, 3, 1, 4, E-2]`	`[-, 6, 0, 2, E21]`	5	210
P1000	`[+, 314, E-2]`	`[-, 602, E21]`	3	1100
B1999	`[314, E-2]`	`[-602, E21]`	2	2000
FP15	`[FP314/-2]`	`[FP-602/21]`	1	30000

F. Charton. Linear Algebra with Transformers. arXiv:2112.01898 [cs.LG].

We propose a new numerical encoding scheme that **uses just a single token** and renders a language model **end-to-end continuous.**



xVal

A Continuous Number Encoding for LLMs

stat.ML arXiv:2310.02989

Project led by Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti

Accepted contribution at the NeurIPS 2023 AI4Science Workshop

Polymathic

“Electron mass = 0.511 MeV”

1. Convert to text & extract numerical values:

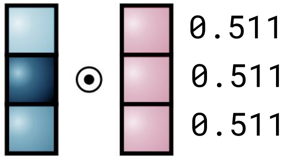
`“Electron mass = [NUM] MeV”`

2. Tokenize:

`[0] [150] [38] [10] [5] [971] [1]`

3. Multiply numerical values into each [NUM] embedding:

`[0] [150] [38] [10] [5] [971] [1]`



Tokenizer Dictionary

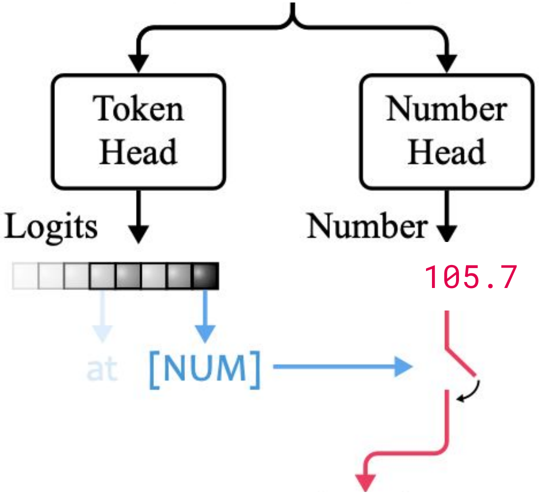
“ → [0]
” → [1]
…
[NUM] → [5]
…
= → [10]
Mass → [38]
Electron → [150]
MeV → [971]

During inference, we can use a mask ([???]) to tell the network where we want to guess a token.

“Muon mass = [???] MeV”

[0] [151] [38] [10] [6] [971] [1]

If a [NUM] token is predicted, the model also prompts a dedicated number transformer head to predict its value.



“Muon mass = 105.7 MeV”

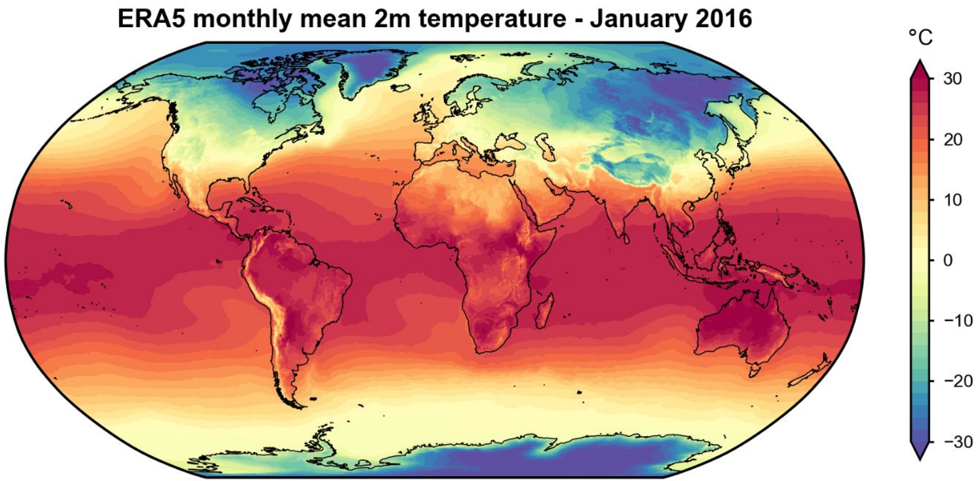
Tokenizer Dictionary

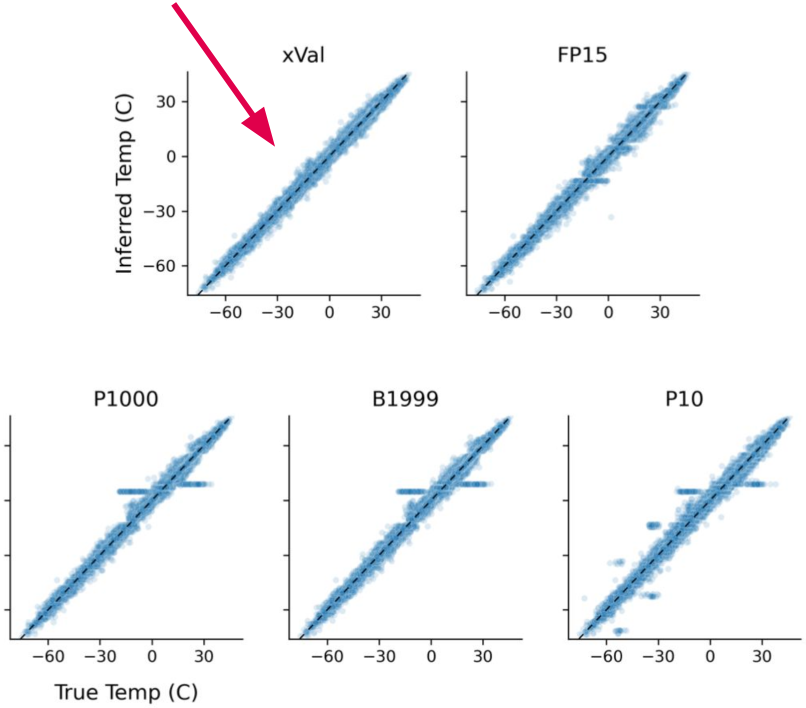
“ → [0]
” → [1]
…
[NUM] → [5]
[???] → [6]
…
= → [10]
Mass → [38]
Electron → [150]
Muon → [151]
MeV → [971]

This encoding strategy has 3 main benefits:

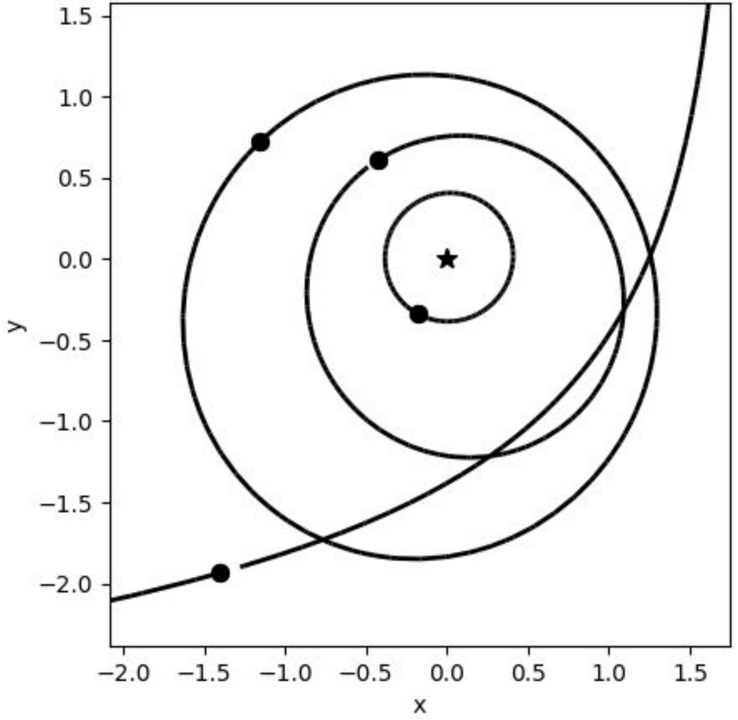
- Continuity
  - It embeds key information about how numbers continuously relate to one another, making its predictions **more appropriate for many scientific applications.**
- Interpolation
  - It makes **better out-of-distribution predictions** than other numerical encodings.
- Efficiency
  - By using just a single token to represent any number, it **requires less memory, compute resources, and training time** to achieve strong results.

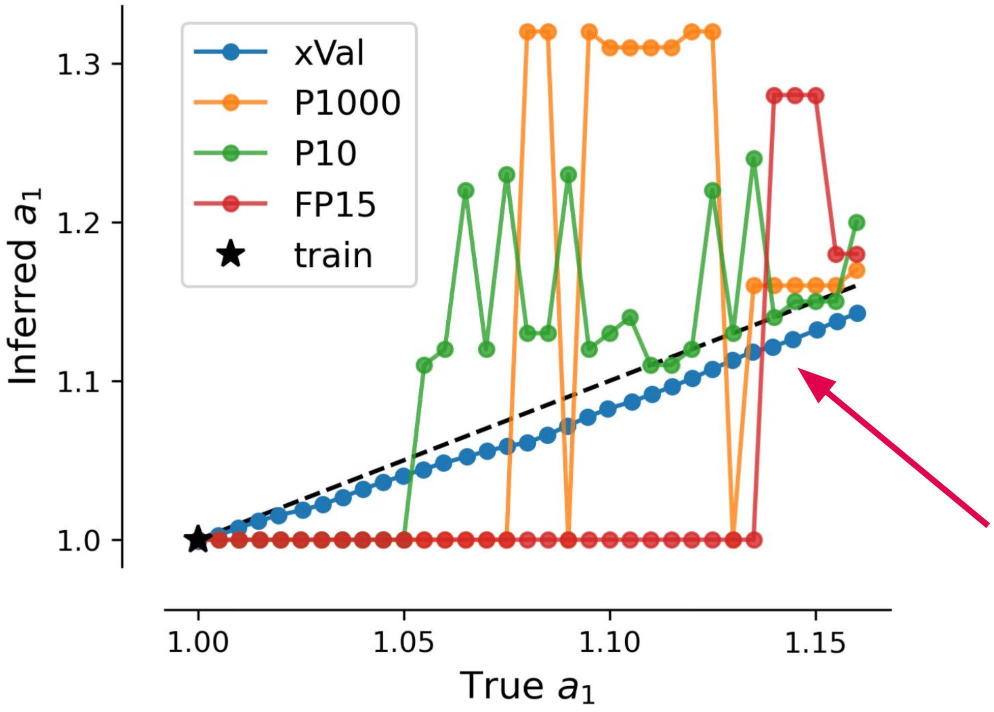
When tested on the task of temperature forecasting from a real-world dataset, xVal achieves the **lowest loss with the shortest training time.**





When tested on the task of extracting orbital data from **simulated planetary motion**, xVal shows improved out-of-distribution predictions.



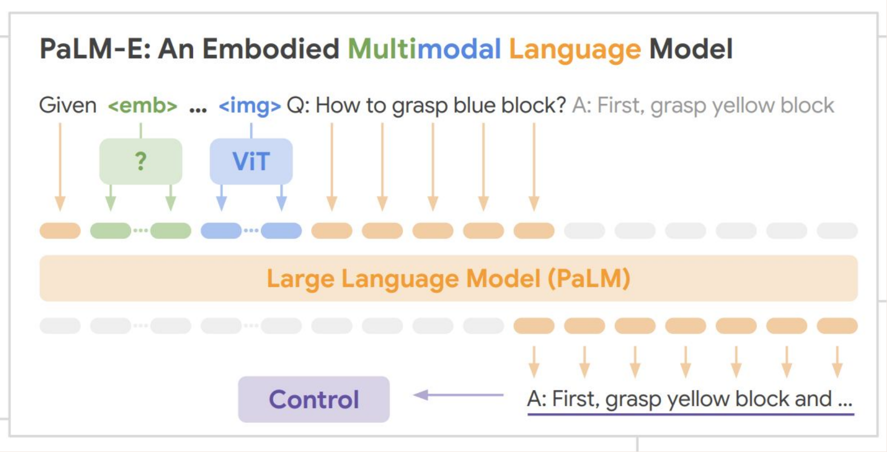


Part 2
Challenges

Challenge I: Inefficiency

Potential Solution: Modality specific coding

- Removing all structure and turning the problem into textual tokens means that the model will have to learn these structures itself. (Very data hungry)



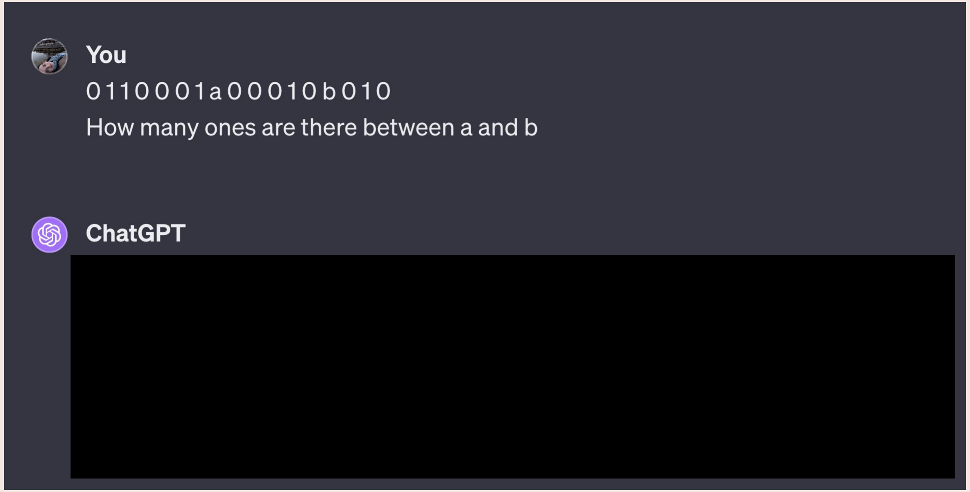
Challenge II:

Time for a few rounds of…

Can GPT4 even do that?!

Question 1.

Count the ones (simple version)



- Success rate is ~70%.
- The task is easy, GPTs (causal autoregressive models) can in−principle solve this efficiently.

Question 1.

Count the ones (simple version)

- Count the number of ones between a and b:

  "1 0 1 0 1 a 0 1 1 1 b 0 0 0 1"

- Easy for transformer with causal mask (and ⟨BOS⟩ token)

Question 2.

Count the ones (select and count task)

Ask me about their circuits!

- Can a transformer based model solve the following problem?
- Count the number of ones between the last a,b pair:

"1 0 1 1 0 1 a 0 1 0 b 1 0 1 0 1 a 0 1 1 1 b 0 0 0 1"

T

"Muon mass = [???] MeV"

suggestions based on LLM lore

The structure of attention is the main cause.

Transformers have highly constrained communication protocols.

- In an attention layer, any token can attend to any other token based on:

  1. their position (absolute or relative) 2. their content

Consequences I

Deserialization is a challenge

- These limitations of transformers imply that the network will be unable to recover the original data from the serialized form.

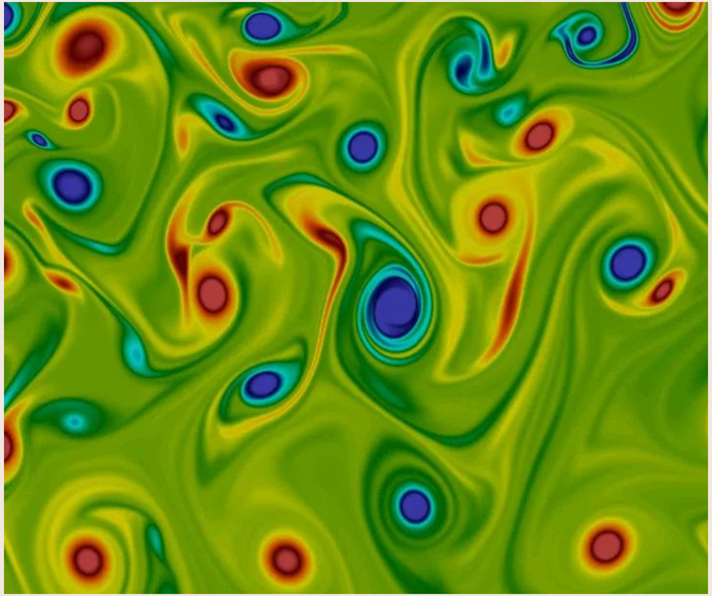
“[[a, b, c], [d, e, f]]” $\rightarrow \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$

- This means that the network will be unable to perform some matrix operations.
- Providing row/column position code fixes this specific problem but other problems remain. (e.g. pooling information from two observations with different resolutions)

Consequences II:

Real world implications for transformer based models

- Take a 2D material with different phases and dynamic boundaries:
- Can we easily calculate the ratio of + to − in each phase?



Conclusions

Building a **foundation model for physics** could be our best shot at addressing the remaining unanswered mysteries in our field. But **science data is heterogeneous**, with many different modalities, and that will make building a foundation model challenging.

The way forward is to somehow **embed each modality into the same embedding space.**

- The **simplest way is converting everything to text**, though ultimately we'll likely want to combine this with other modality-specific embedding strategies.

Let's say we convert everything to text.

- **Handling numbers in a continuous way is a challenge.** A numerical encoding like xVal could help adapt some of the key structures of LLMs to make them more appropriate for scientific analysis.
- Moreover, **there are limits to what operations transformers can perform**, especially under default configurations that have been deemed best for natural language processing.

Thanks!

Some good questions you could ask us...

"Why not just let ChatGPT use a calculator?"

"How does xVal represent really big/small numbers?"

The internal representation of x is bounded by layer normalization...

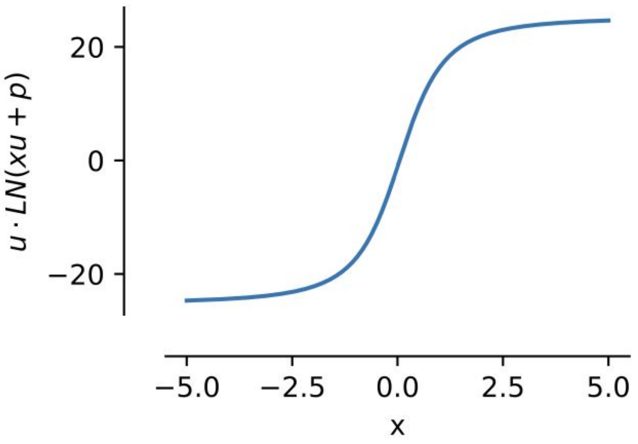
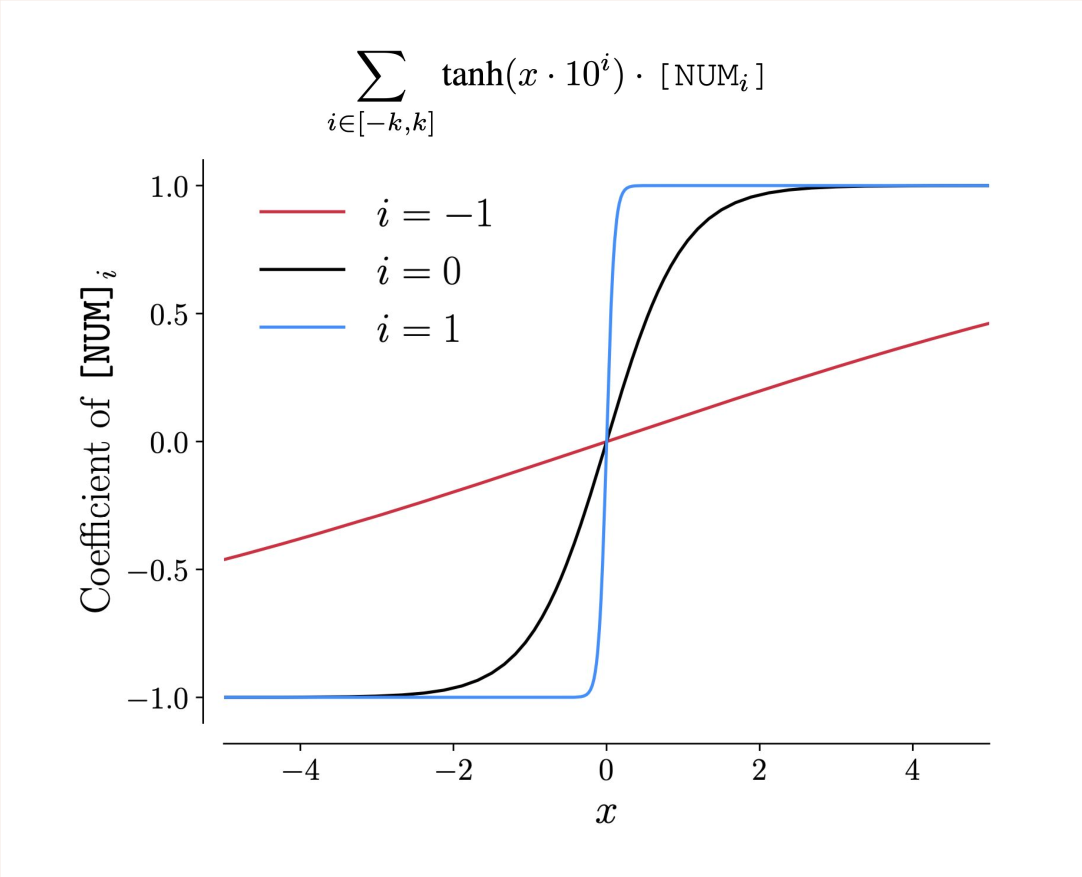


Figure 2: Value of the embedding of the number x after layer-norm, projected onto the direction of the `[NUM]` embedding vector.

...but multiple numerical embeddings can be used to capture a wider range of values:



Using scientific notation, e.g. writing numbers as e.g. 832 → “8 10e2 3 10e1 2 10e0”, is somewhat helpful, but the model still isn’t learning the basic rules of arithmetic.

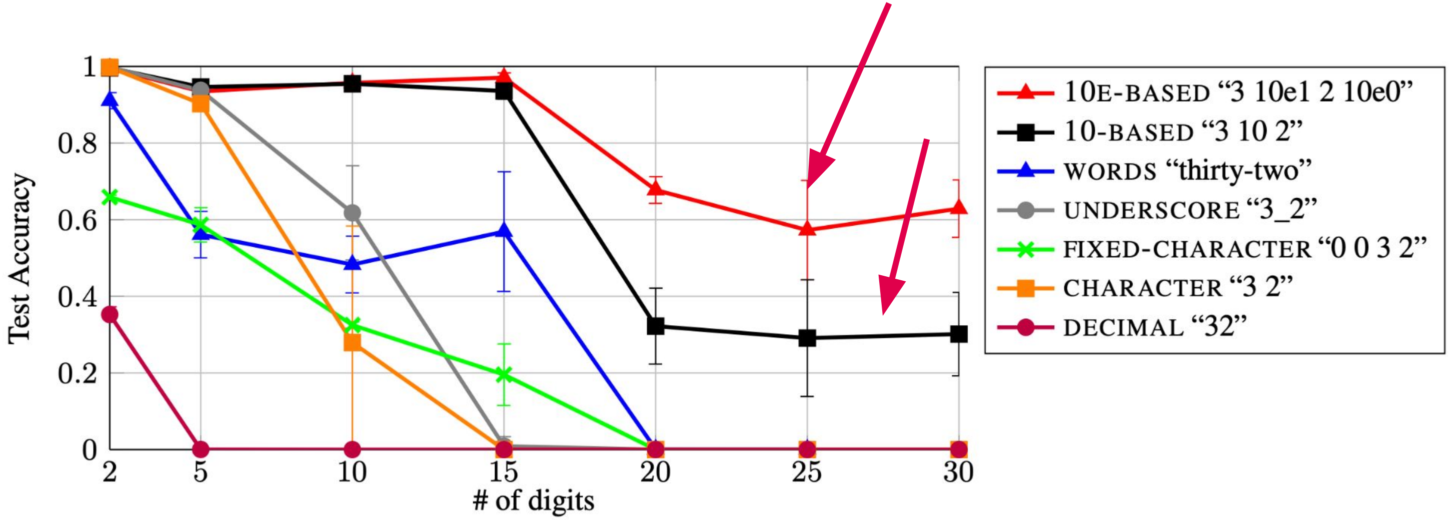


Figure 1: Accuracy of different number representations on the addition task.

Investigating the Limitations of Transformers with Simple Arithmetic Tasks. arXiv:2102.13019 [cs.CL].

Another strategy:
mapping numbers onto a basis of "prototype numerals".

For a numeral n, we will first transform it into a scientific notation and determine its exponent $e(n) \in \mathbb{Z}$ and mantissa $f(n) \in (-10, 10)$, as shown in Equation 1.

$$n = 10^{e(n)} \times f(n). \tag{1}$$

For example, -123 can be transformed to -1.23×10^2. Its exponent, denoted as $e(-123)$, is the 2, and its mantissa, denoted as $f(-123)$, is -1.23.

Embed the exponent as a vector associated with integers between -8 and +12

Embed the mantissa as a sum of distances from "prototype numerals" distributed uniformly between [-10, 10]:

$$\mathrm{NE}_i^f(f(n)) = \exp\left(-\frac{\|f(n) - q_i^f\|^2}{\sigma^2}\right)$$

NumGPT: Improving Numeracy Ability of Generative Pre-trained Models. arXiv:2109.03137 [cs.CL].

Table 2: Adjusted R^2 scores calculated between predictions and true values for the different encodings on various arithmetic datasets. (Higher is better; $R^2 = 1$ is the theoretical maximum.)

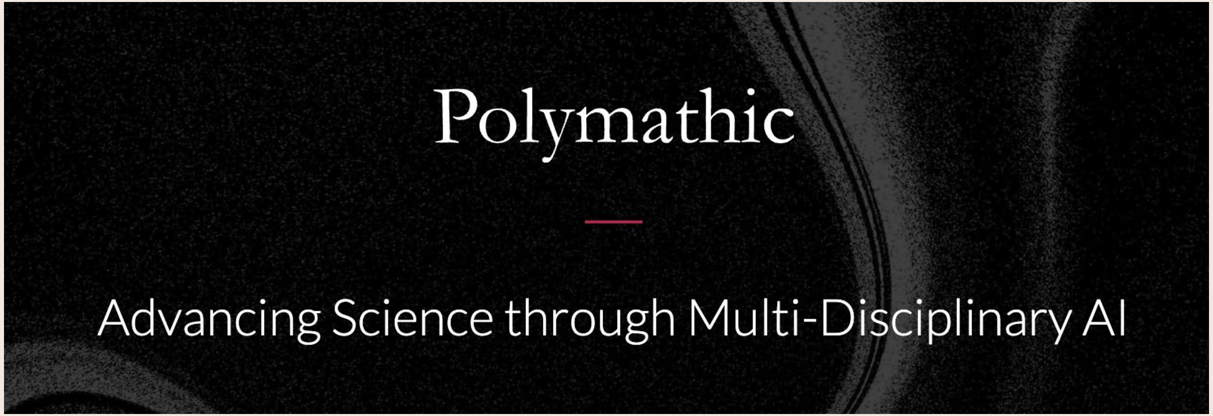
Encoding	3-digit Multiplication	4-digit Multiplication	5-digit Multiplication
P10	0.9989	0.6071	0.9439
P1000	0.9997	0.9783	0.9991
B1999	0.9998	0.9984	0.9997
FP15	0.7119	0.9959	0.9980
xVal	0.9986	0.9975	0.9958

Table 3: Arithmetic evaluation task of random binary trees combining different numbers of operands with addition, subtraction, and multiplication. R^2 measured between true expression value and transformer prediction (scores in parentheses include outliers). P10 evaluated on the 4-operands dataset did not converge within our allocated compute time.

Encoding	2 operands	3 operands	4 operands
P10	0.998	0.996	N/A
P1000	0.991	0.990	0.991
FP15	0.993	0.981	0.935
xVal	0.99998	0.99994	0.99998

Polymathic AI Collaboration

polymathic-ai.org



SIMONS FOUNDATION

NYU

UNIVERSITY OF CAMBRIDGE

SCHMIDT FUTURES

PRINCETON UNIVERSITY

BERKELEY LAB

Polymathic AI Collaboration

polymathic-ai.org