Where are we headed in HPC + AI for Scientific Discovery

Rick Stevens
Argonne National Laboratory
The University of Chicago

Crescat scientia; vita excolatur

What is the most important application for Exascale Computers?

What is the most important application for Exascale computers?

The most important application for Exascale computers is to provide a platform for scientific research that can lead to new discoveries in a variety of fields.

GPT-3 “DaVinci-002”

Write a poem on high-performance computing in the style of Robert Frost.

GPT-3 “DaVinci-002”

The computing world is full of speeds and feeds
But at the end of the day
We just want our code to run
At the highest performance possible

Our machines are designed for this
With massive cores and GPUs
We have the power to crunch numbers
Faster than ever before

But there's a limit to this power
As we all know
No matter how fast our machines
Our code will always be slow

There's always room for improvement
And we'll keep pushing
To make our code run faster
On the machines we have today

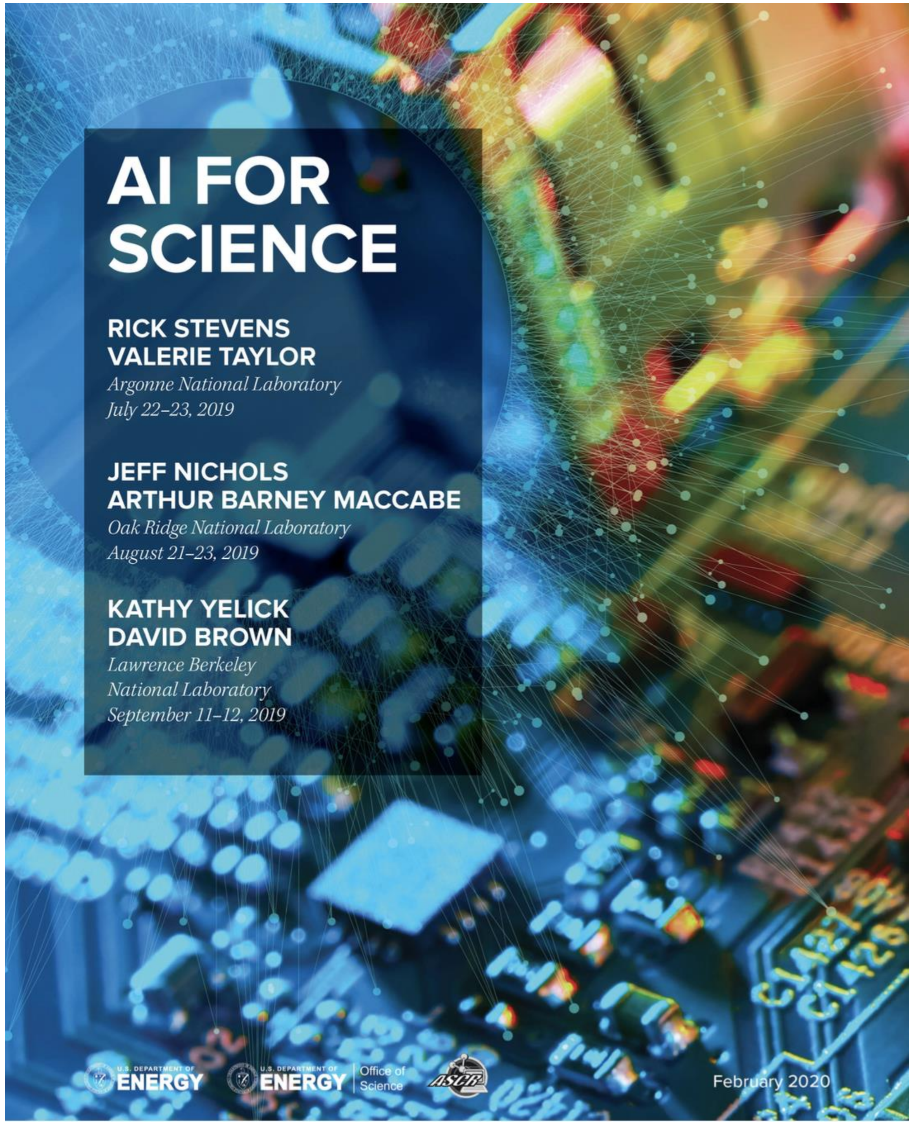
Directional Trends (HPC + AI for Science)

- **GPUs are winning even though its still modest fractions of production codes are well tuned for current hardware**
  - Opportunity for AI driven code conversions and standard programming model
- **GPUs are becoming somewhat more general purpose and simultaneously more optimized for AI workloads**
  - Low precision support (new FP8, tensor support (xGEMM)
- **Foundation Models are getting more interesting**
  - Future is probably going to be small numbers of very powerful general-purpose models that get specialized (!) for downstream applications
- **AI Surrogates for traditional HPC applications will have enormous impact in the next few years as they catch on**
  - Today's exascale machines will be the places where "effective" zetta and yotta scale applications first happen
- **Some ideas are not going to pan out as originally planned**
  - Neuromorphic platforms (good for computational neuroscience kinda), might find a role in low power edge computing but will unlikely become mainstream replacements for CPUs/GPUs
  - Quantum computing is a long-term play (i.e., 20-50 years), not 3 not 5 probably not 10
  - Specialized AI hardware is important as it pushes new architectural concepts is competing on two fronts 1) GPU flexibility in allocating transistors to tensor ops, etc. 2) Software investment

Directional Trends (HPC + AI for Science)

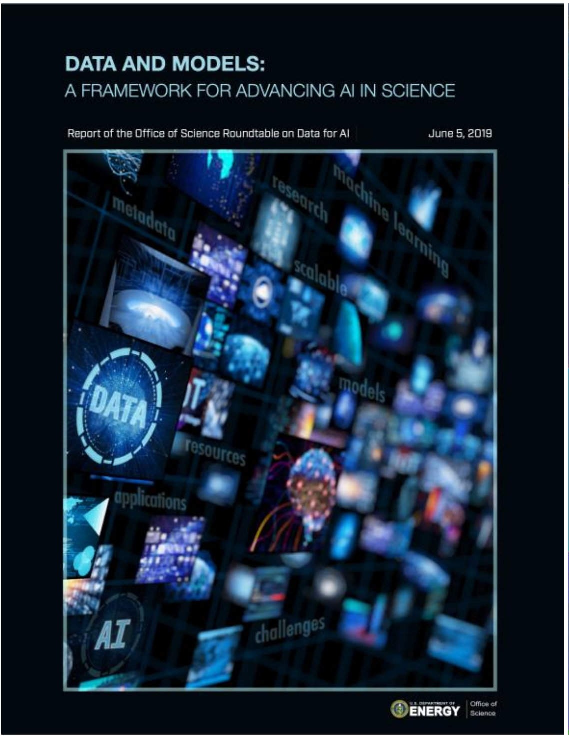
- **AI will have an accelerating impact on programming, software engineering, code optimization and tuning**
  - Current tools are just the beginning
- **X as a Service (XaaS) concepts will get richer and more interesting**
  - AI as a Service (e.g. NeMO), Workflows as a Service, Surrogates as a Service?
- **Chip Power efficiency is now a political issue**
  - Companies pledge to improve power efficiency but of course politics is not science, we need to actually have technical means to do so!
- **If performance is your goal, you can't ignore AI and ML approaches**
  - Near linear sorting and searching by learning on real distributions
- **Future apps might look weird to us (mixed precision and ML surrogates)**
  - Massive low precision capabilities (Zetta ops by 2030) perhaps earlier
- **FPGAs promising for some applications, but the software stacks still suck**
  - Now that major players are owned by CPU/GPU companies this might improve

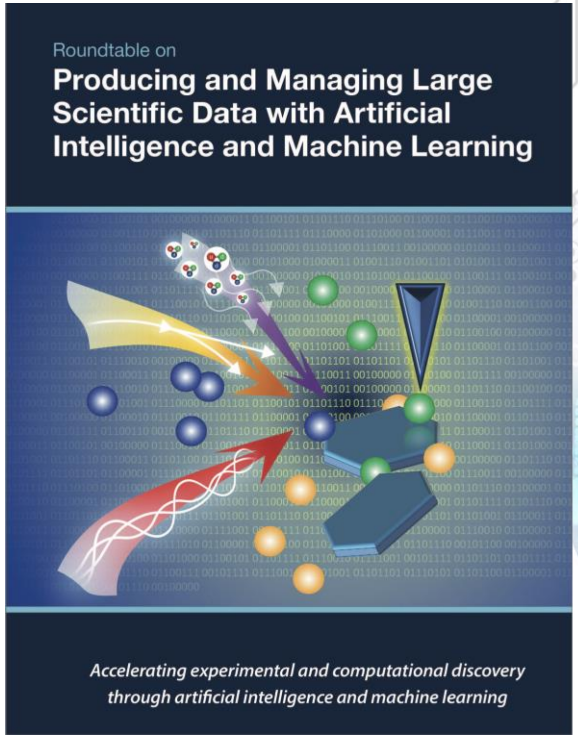
AI for Science – What's Next After Exascale



- Over 1,300 scientists participated in four town halls during the summer/fall of 2019
- Research Opportunities in AI
  - Biology, Chemistry, Materials,
  - Climate, Physics, Energy, Cosmology
  - Mathematics and Foundations
  - Data Life Cycle
  - Software Infrastructure
  - Hardware for AI
  - Integration with Scientific Facilities
- Modeled after the Exascale Series in 2007
- ASCAC subcommittee Report Sept 2020

https://www.anl.gov/ai-for-science-report





U.S. DEPARTMENT OF ENERGY | Office of Science

Artificial Intelligence for Earth System Predictability

AI4ESP

BER-ASCR planning project to create a paradigm shift in the science of predictability and prediction, based on the co-design of artificial intelligence methodologies with physics-based observing, data assimilation, modeling, and prediction.

To be effective, DOE wants AI4ESP as a collaboration - as an extension beyond MODEX to more rapidly facilitate co-design and more rapidly reduce uncertainties of predictions

Nicki L. Hickmon - PI
Argonne National Laboratory

Scott M. Collis
Argonne National Laboratory

Forrest M. Hoffman
Oak Ridge National Laboratory

Haruko M. Wainwright
Lawrence Berkeley National Laboratory

Summer 2022 Workshop Topics

June, July, August

Argonne, Berkeley, Oak Ridge, Livermore, Los Alamos, Sandia

- AI for Advanced Properties Inference and Inverse Design
- AI and Robotics for Autonomous Discovery
- AI Based Surrogates for HPC
- AI for Programming and Ssoftware Engineering
- AI for Prediction and Control of Complex Engineered Systems
- Foundation AI for Scientific Knowledge Discovery, Integration and Synthesis

Exascale Target Platforms

Intel CPU/GPUs, AMD CPU/GPUs

These architectures will excel at AI problems and AI + HPC problems

Argonne

Oak Ridge

Livermore

Foundation AI for Scientific Discovery, Knowledge Synthesis and Integration

- LLMs appear to be very general sequence-oriented models that learn from self-supervised examples
- Anything that can be tokenized to sequences is fair game
  - Text, Code, DNA, RNA, Proteins, Protocols, Graphs (triples)
  - Images (tokenized as patches), Waveforms (tokenized as samples)
  - Robotic control sequences
  - Time dependent data
- Large-scale Models trained on broad data, can be specialized for downstream tasks with much less work than models trained from scratch
- Many many tasks in biology and life sciences can be addressed with LLMs, from knowledge distillation from literature, to generating sequences (proteins, viruses, microbes), to small molecules, to protocols
- LLMs can also generate mathematical solutions to problems

https://arxiv.org/pdf/2108.07258.pdf

GPT-3, T5, M6, Gopher, etc. >> 100B parameter models

On the Opportunities and Risks of Foundation Models

Rishi Bommasani* Drew A. Hudson Ehsan Adeli Russ Altman Simran Arora
Sydney von Arx Michael S. Bernstein Jeannette Bohg Antoine Bosselut Emma Brunskill
Erik Brynjolfsson Shyamal Buch Dallas Card Rodrigo Castellon Niladri Chatterji
Annie Chen Kathleen Creel Jared Quincy Davis Dorottya Demszky Chris Donahue
Moussa Doumbouya Esin Durmus Stefano Ermon John Etchemendy Kawin Ethayarajh
Li Fei-Fei Chelsea Finn Trevor Gale Lauren Gillespie Karan Goel Noah Goodman
Shelby Grossman Neel Guha Tatsunori Hashimoto Peter Henderson John Hewitt
Daniel E. Ho Jenny Hong Kyle Hsu Jing Huang Thomas Icard Saahil Jain
Dan Jurafsky Pratyusha Kalluri Siddharth Karamcheti Geoff Keeling Fereshte Khani
Omar Khattab Pang Wei Koh Mark Krass Ranjay Krishna Rohith Kuditipudi
Ananya Kumar Faisal Ladhak Mina Lee Tony Lee Jure Leskovec Isabelle Levent
Xiang Lisa Li Xuechen Li Tengyu Ma Ali Malik Christopher D. Manning
Suvir Mirchandani Eric Mitchell Zanele Munyikwa Suraj Nair Avanika Narayan
Deepak Narayanan Ben Newman Allen Nie Juan Carlos Niebles Hamed Nilforoshan
Julian Nyarko Giray Ogut Laurel Orr Isabel Papadimitriou Joon Sung Park Chris Piech
Eva Portelance Christopher Potts Aditi Raghunathan Rob Reich Hongyu Ren
Frieda Rong Yusuf Roohani Camilo Ruiz Jack Ryan Christopher Ré Dorsa Sadigh
Shiori Sagawa Keshav Santhanam Andy Shih Krishnan Srinivasan Alex Tamkin
Rohan Taori Armin W. Thomas Florian Tramèr Rose E. Wang William Wang Bohan Wu
Jiajun Wu Yuhuai Wu Sang Michael Xie Michihiro Yasunaga Jiaxuan You Matei Zaharia
Michael Zhang Tianyi Zhang Xikun Zhang Yuhui Zhang Lucia Zheng Kaitlyn Zhou
Percy Liang*[1]

Center for Research on Foundation Models (CRFM)
Stanford Institute for Human-Centered Artificial Intelligence (HAI)
Stanford University

AI is undergoing a paradigm shift with the rise of models (e.g., BERT, DALL-E, GPT-3) that are trained on broad data at scale and are adaptable to a wide range of downstream tasks. We call these models foundation models *to underscore their critically central yet incomplete character. This report provides a thorough account of the opportunities and risks of foundation models, ranging from their capabilities (e.g., language, vision, robotics, reasoning, human interaction) and technical principles (e.g., model architectures, training procedures, data, systems, security, evaluation, theory) to their applications (e.g., law, healthcare, education) and societal impact (e.g., inequity, misuse, economic and environmental impact, legal and ethical considerations). Though foundation models are based on standard deep learning and transfer learning, their scale results in new emergent capabilities, and their effectiveness across so many tasks incentivizes homogenization. Homogenization provides powerful leverage but demands caution, as the defects of the foundation model are inherited by all the adapted models downstream. Despite the impending widespread deployment of foundation models, we currently lack a clear understanding of how they work, when they fail, and what they are even capable of due to their emergent properties. To tackle these questions, we believe much of the critical research on foundation models will require deep interdisciplinary collaboration commensurate with their fundamentally sociotechnical nature.*

[1]Corresponding author: pliang@cs.stanford.edu *Equal contribution.

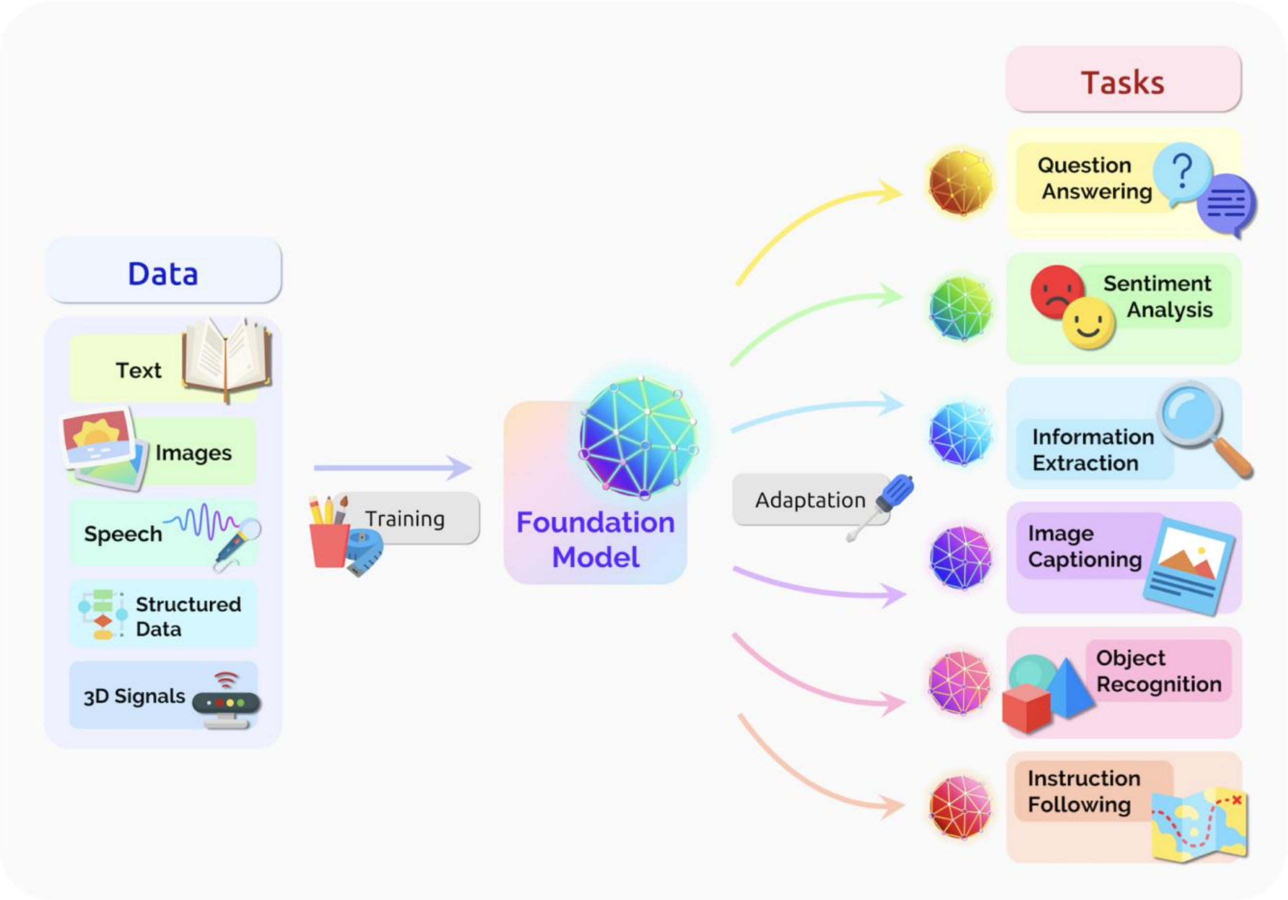
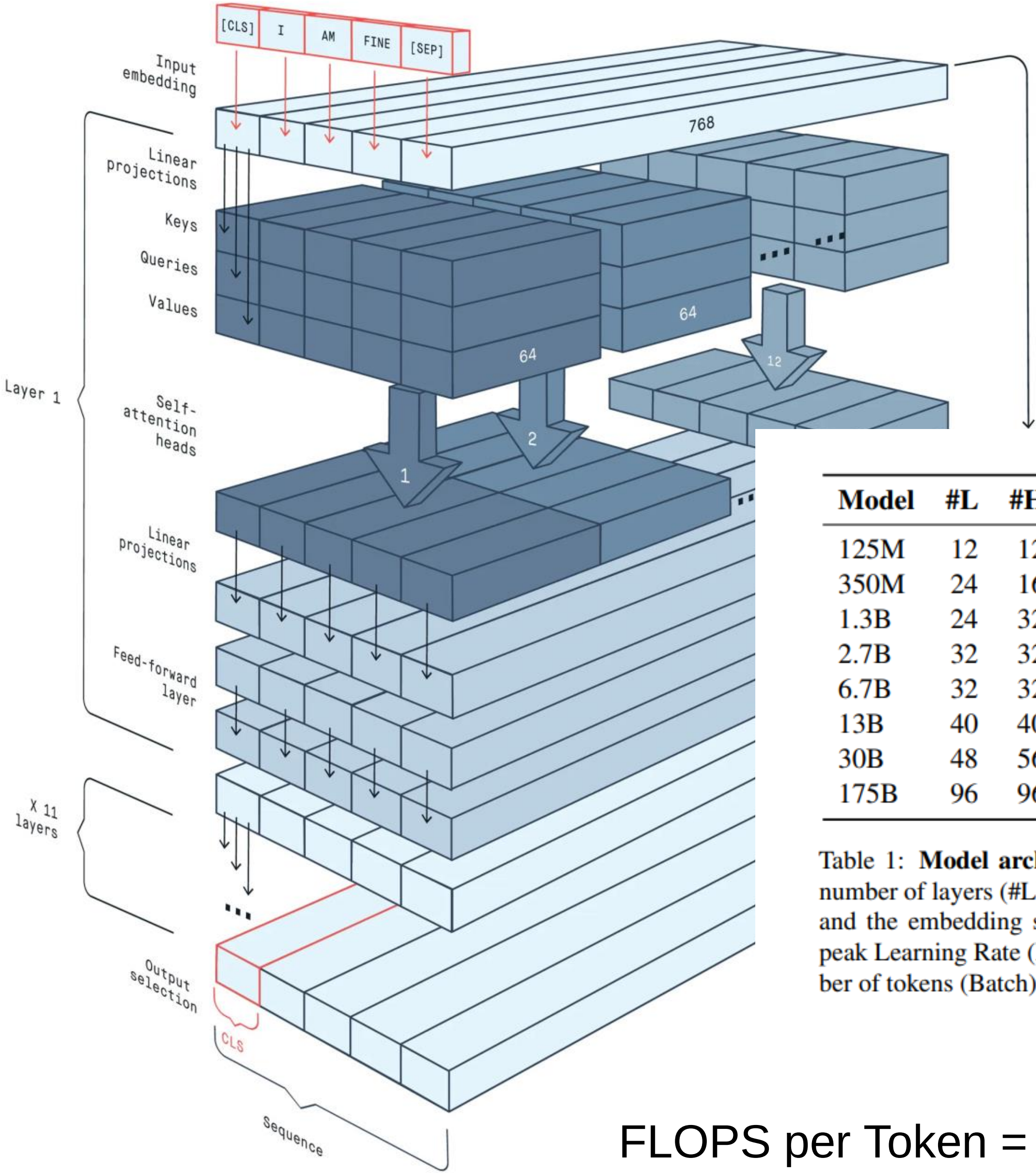


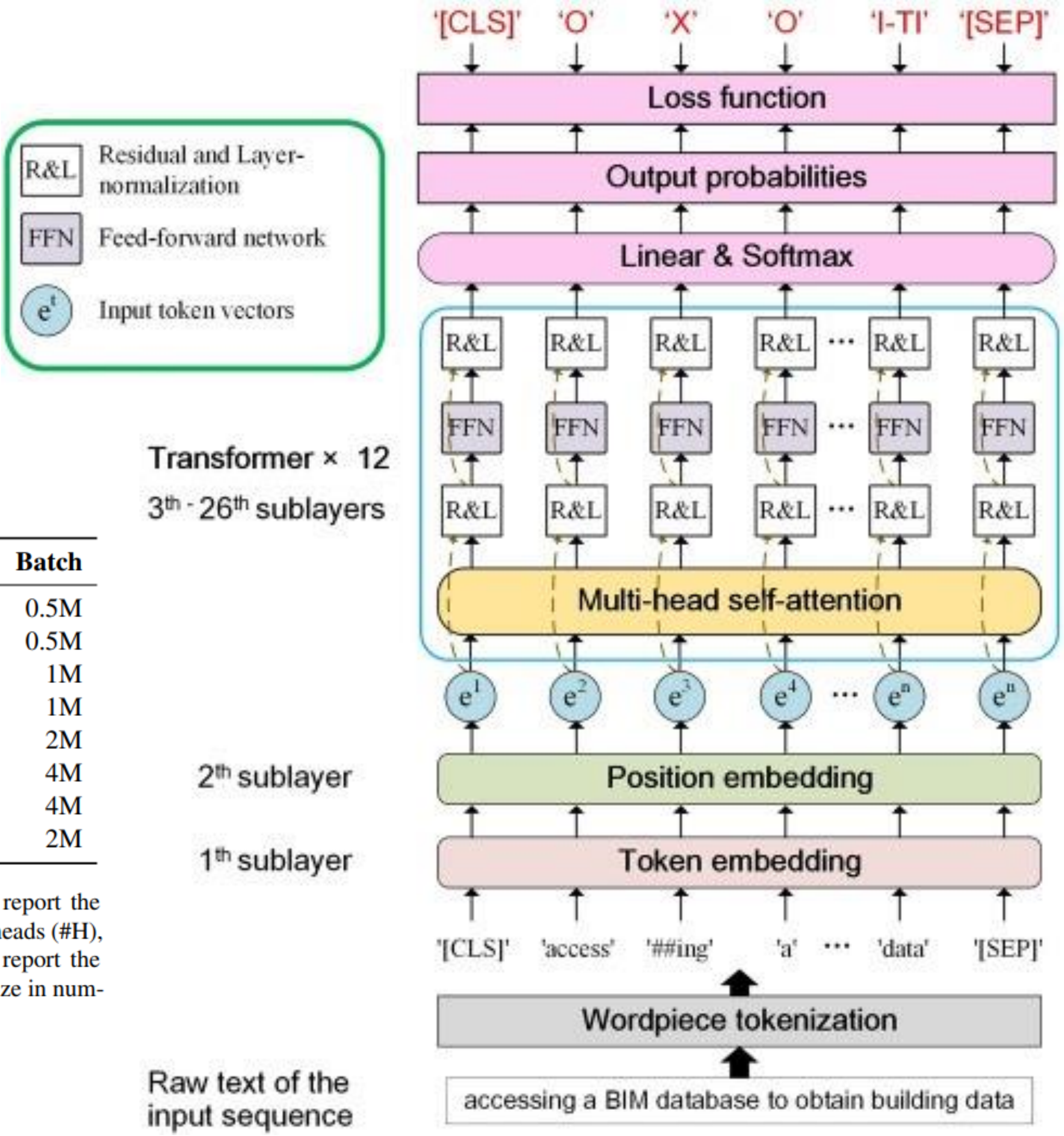
Fig. 2. A foundation model can centralize the information from all the data from various modalities. This one model can then be adapted to a wide range of downstream tasks.

The Transformer with Self-Attention



Model	#L	#H	d_{model}	LR	Batch
125M	12	12	768	6.0e−4	0.5M
350M	24	16	1024	3.0e−4	0.5M
1.3B	24	32	2048	2.0e−4	1M
2.7B	32	32	2560	1.6e−4	1M
6.7B	32	32	4096	1.2e−4	2M
13B	40	40	5120	1.0e−4	4M
30B	48	56	7168	1.0e−4	4M
175B	96	96	12288	1.2e−4	2M

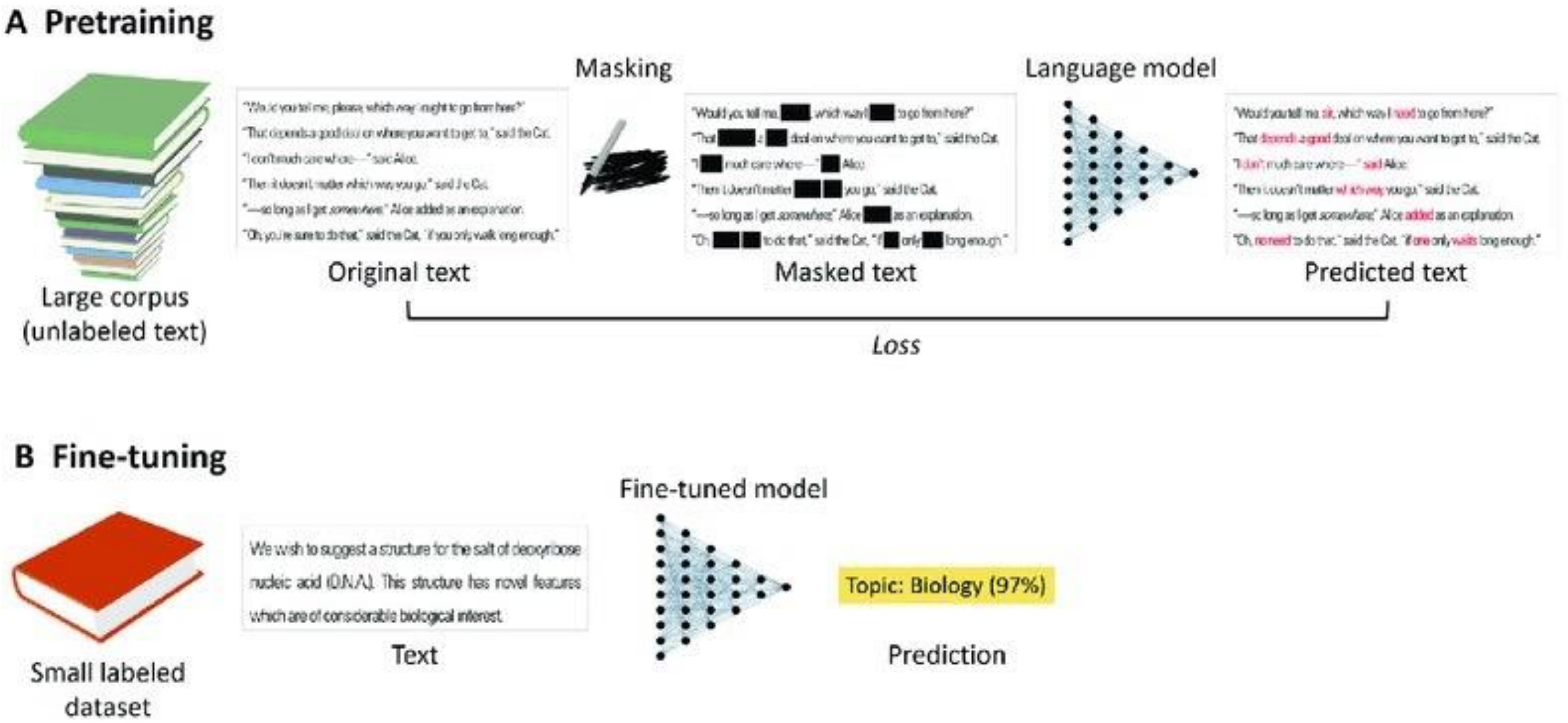
Table 1: **Model architecture details.** We report the number of layers (#L), number of attention heads (#H), and the embedding size (d_{model}). We also report the peak Learning Rate (LR) and global batch size in number of tokens (Batch).



FLOPS per Token = 6N + 12LHQT

Layers, Heads, Dimension, Width

Self-Supervised Pretrained Autoregressive Models



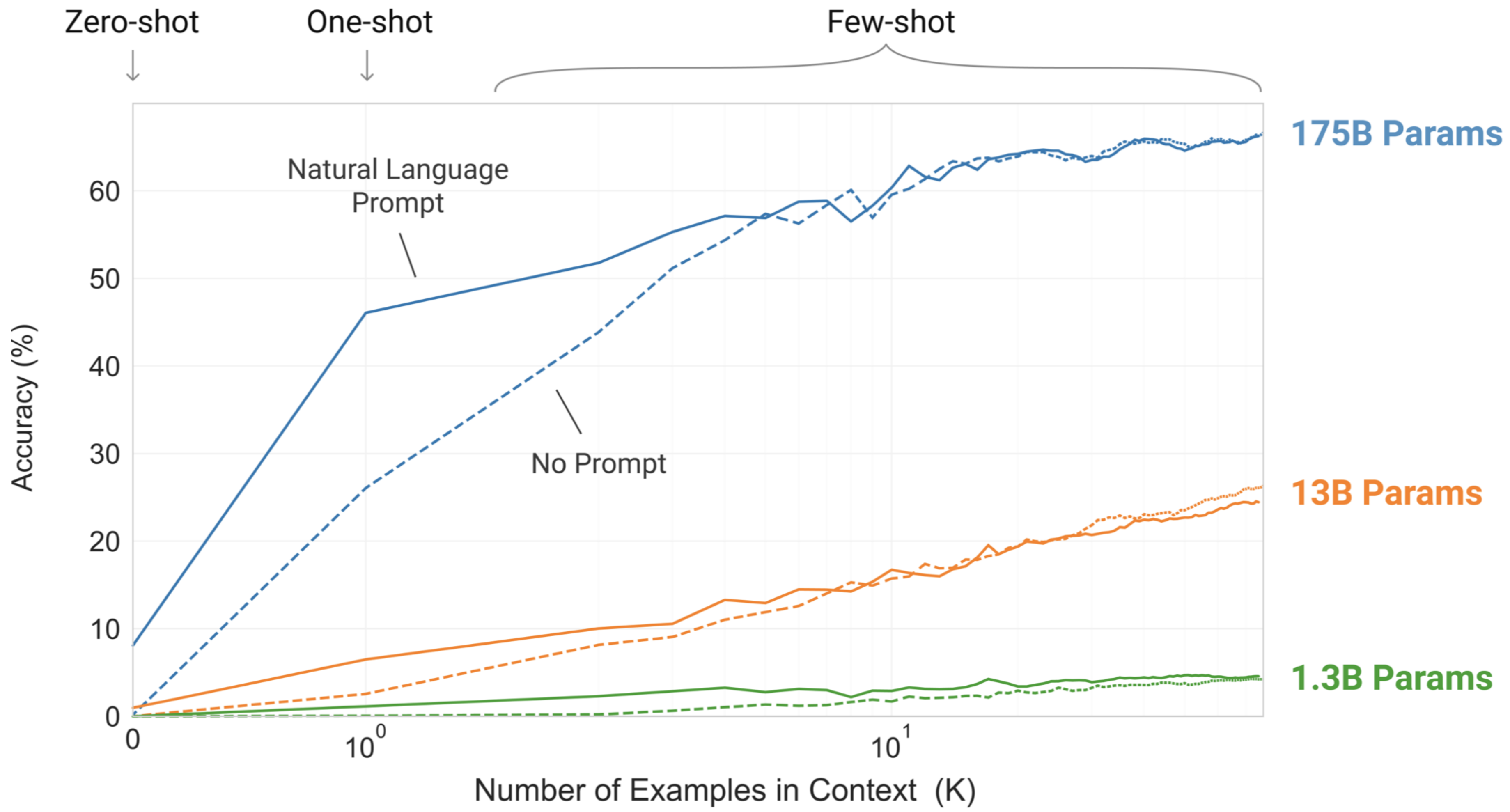


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

arXiv:2005.14165 Brown et. al. "Language Models are Few Shot Learners"

Emergence from Scale and Prompting

Foundation models have also led to surprising emergence which results from scale. For example, GPT-3 [Brown et al. 2020], with 175 billion parameters compared to GPT-2's 1.5 billion, permits in-context learning, in which the language model can be adapted to a downstream task simply by providing it with a prompt (a natural language description of the task), an emergent property that was neither specifically trained for nor anticipated to arise.

What are the step by step directions for making beer?

1. Preheat the oven to 375 degrees Fahrenheit.
2. In a pot, bring to life a cup of fresh brewing water.
3. Pour this water into the cup and stir to combine.
4. Place the cup into the oven and away you go!
5. Bake the beer for a specific amount of time, typically about 30 minutes.
6. Once the beer is done, remove it from the oven and let it cool.
7. Once it's out of the oven, let it cool on the cutting board for about 2 minutes before serving.
8. Serve the beer cold or add a little bit of ice to it before serving.

GPT-3 with 1B Parameters

What are the step by step directions for making beer?

1. Choose a recipe.
2. Sanitize all of your brewing equipment.
3. Prepare your ingredients.
4. Mash the grains.
5. Boil the wort.
6. Cool the wort.
7. Transfer the wort to a fermenter.
8. Add yeast to the fermenter.
9. Allow the beer to ferment.
10. Package the beer.

GPT-3 with 175B Parameters

Wu Dao --- Mixture of Experts 10Tp

9项
精准纪录

国际公认Benchmark
取得优异成绩

ImageNet
zero-shot SOTA
200类超越 OpenAI CLIP

LAMA
知识探测
超过 AutoPrompt

LAMBADA
完形填空
超过1.7倍参数微软 Turing NLG

SuperGLUE few-shot
FewGLUE
超越 OpenAI GPT-3

UC Merced Land-Use
zero-shot SOTA
超越 OpenAI CLIP

MSCOCO
文生成图
超越130亿参数 OpenAI DALL·E

MSCOCO
英文图文检索
超过OpenAI CLIP，Google ALIGN

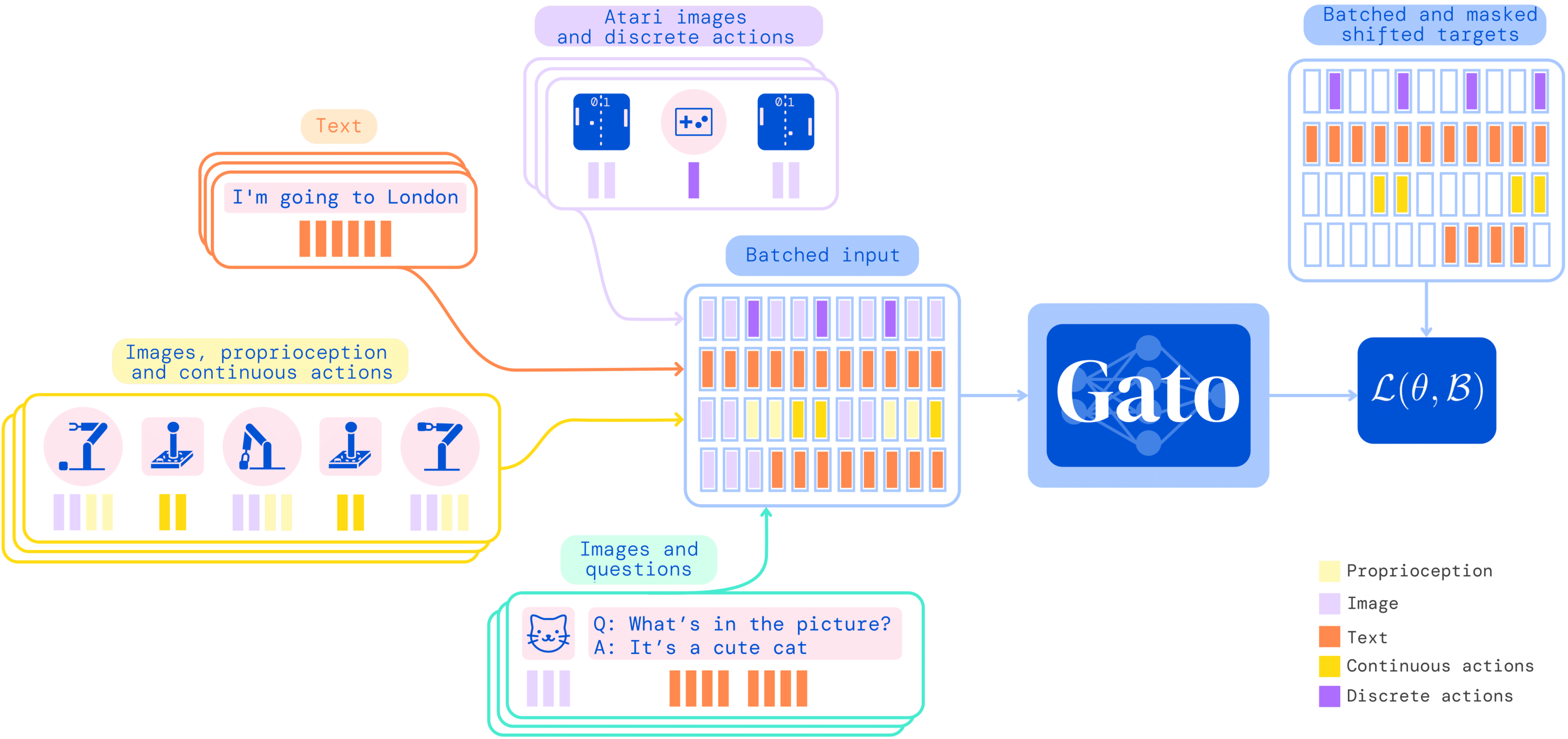
MSCOCO
多语言图文检索
超越 UC2, M3P

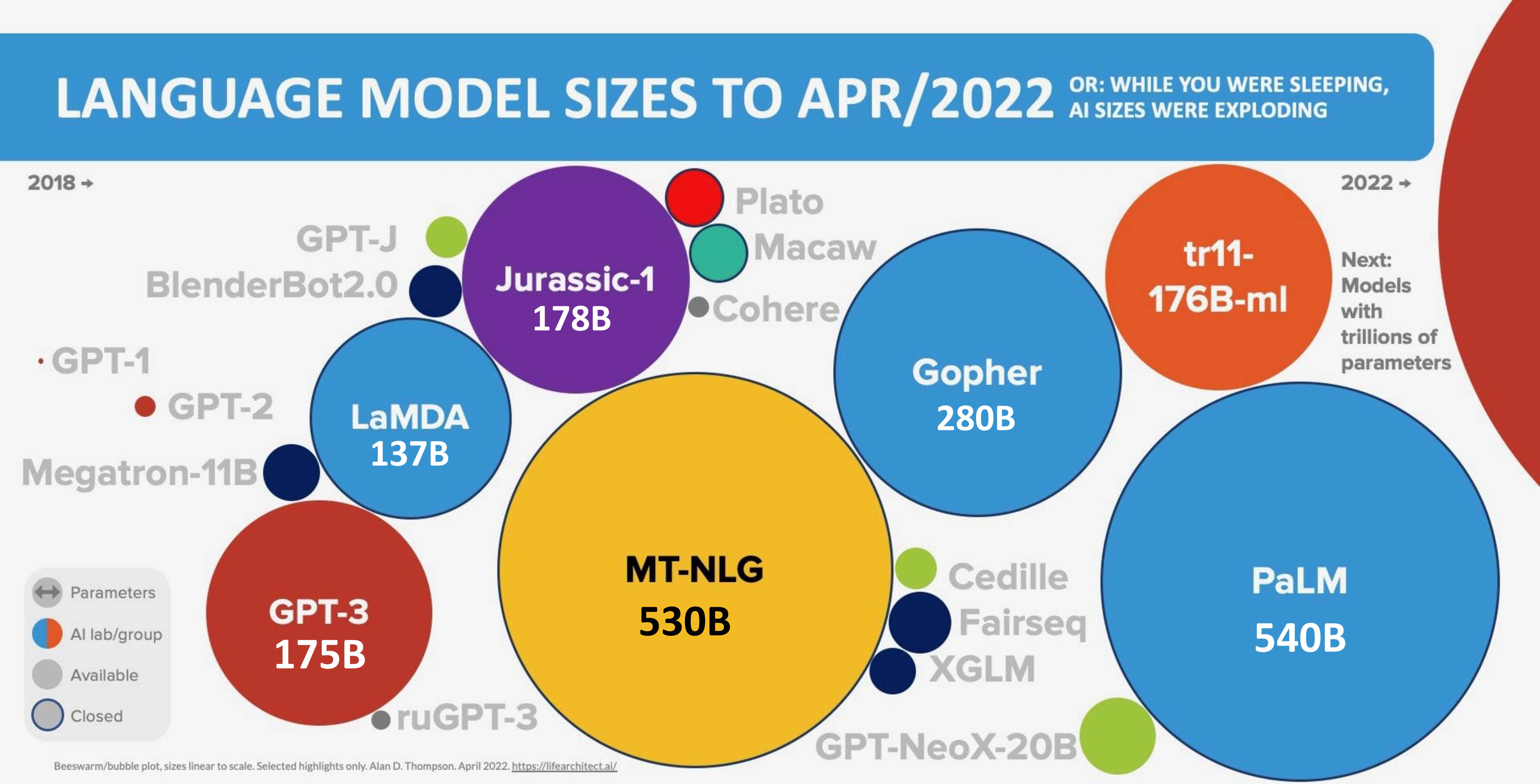
Multi 30K
多语言图文检索
超越 UC2, M3P

Towards fewer more powerful models

- Transfer, Multitask, and Multimodal Learning
  - Learning more than one task at a time
  - Can often be better at each task than a separate network for each task
  - Networks can learn to share representation and learning across problems
  - Networks have learned up to ~~~50~~ ~~192~~ 600 different tasks
- Very Large Networks have been trained
  - Networks with ~~billions~~ > 10 trillions of weights (100 trillion in plan)
  - Can absorb huge datasets and still improve with more data (LLMs)

DeepMind’s GATO 600 tasks – one model



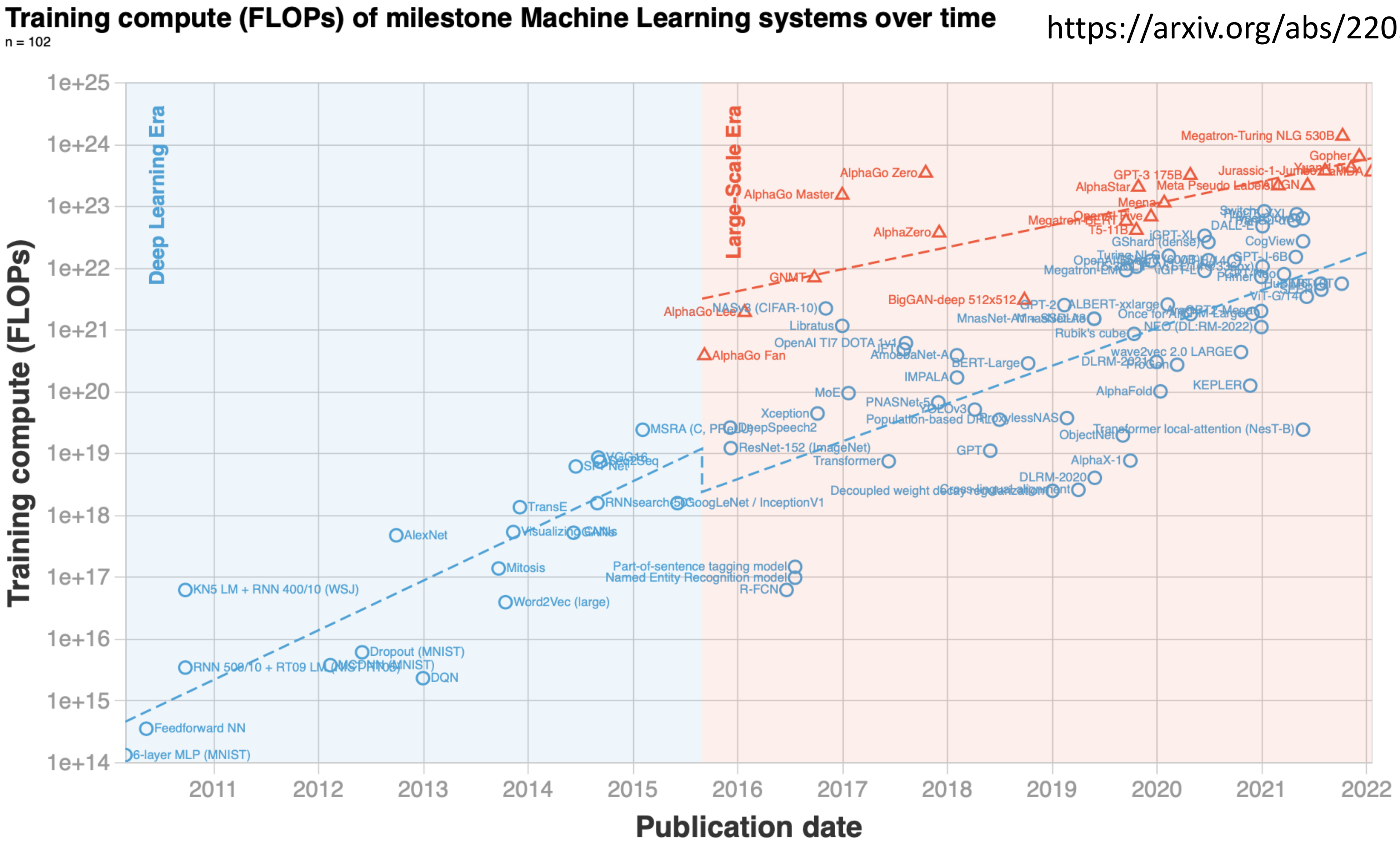


LANGUAGE MODEL SIZES TO APR/2022
OR: WHILE YOU WERE SLEEPING, AI SIZES WERE EXPLODING
2018 →
2022 →
GPT-J
BlenderBot2.0
GPT-1
GPT-2
Megatron-11B
Jurassic-1
178B
LaMDA
137B
GPT-3
175B
Plato
Macaw
Cohere
MT-NLG
530B
ruGPT-3
Gopher
280B
Cedille
Fairseq
XGLM
GPT-NeoX-20B
tr11-176B-ml
Next: Models with trillions of parameters
PaLM
540B
Parameters
AI lab/group
Available
Closed
Beeswarm/bubble plot, sizes linear to scale. Selected highlights only. Alan D. Thompson. April 2022. https://lifearchitect.ai/

https://arxiv.org/abs/2202.05924

EF day

EF second



EF day

EF second

Figure 3: Trends in training compute of $n102$ milestone ML systems between 2010 and 2022. Notice the emergence of a possible new trend of large-scale models around 2016. The trend in the remaining models stays the same before and after 2016.

Compute Needed to Train

PaLM 540B was trained for 29 days on 4096 v4 TPUs @ Exaflop/s BFP16

Model	TFLOPs per token		Train FLOPs	PetaFLOP/s-days
	(non-attn+attn)	(non-attn+attn+remat)		
8B	0.0550	0.0561	4.29×10^{22}	497
62B	0.388	0.392	3.08×10^{23}	3570
540B	3.28	4.10	2.56×10^{24}	29600

Table 21: Compute usage to train PaLM 8B and PaLM 540B to 780 billion tokens and PaLM 62B to 795B tokens.

arXiv:2204.02311v2 [cs.CL] 7 Apr 2022

LLMs are getting curiously interesting

~~But are not yet very good at reasoning and math~~

Starting to show some skill in reasoning and math

The Foundation Concept in action

PaLM = Google's 540B LLM

Minerva = PaLM + Math

PaLM Coder = PaLM + Code

AlphaCode

PaLM + Math Content Extended Training + CoT

2.2 Models and Training Procedure

Our approach is to start with the PaLM pretrained decoder-only transformer language models Chowdhery et al. (2022), and further train (finetune) them on our mathematical dataset using an autoregressive objective. Table 2 contains the main model and training hyperparameters. The largest model, with 540B parameters, was finetuned on 26B tokens. While this model is highly undertrained compared to the 8B and 62B models, it still achieves superior performance. Additional details can be found in Appendix C.

Table 2: Model architecture and continued training hyperparameters. Model training was resumed from the pretrained PaLM models, and the number of steps quoted refers only to continued training on our technical dataset.

Model	Layers	Heads	d_{model}	Parameters	Steps	Tokens
Minerva 8B	32	16	4096	8.63B	624k	164B
Minerva 62B	64	32	8192	62.50B	416k	109B
Minerva 540B	118	48	18 432	540.35B	399k	26B

CoT == Chain-of-Thought

arXiv:2206.14858v1 [cs.CL] 29 Jun 2022

Question: A uniform solid disk starts from rest and rolls down an inclined plane without slipping. After some time, what fraction of the disk's total kinetic energy is rotational kinetic energy? What of the following is the right choice? Explain you answer.
(A) 1/4 (B) 1/3 (C) 1/2 (D) 2/3

Model output: The disk has both translational and rotational kinetic energy. Let v be the speed of the center of mass of the disk, I be the moment of inertia of the disk, M be the mass of the disk, and R be the radius of the disk. The translational kinetic energy of the disk is given by

$$K_t = \frac{1}{2}Mv^2.$$

The rotational kinetic energy of the disk is given by

$$K_r = \frac{1}{2}I\omega^2 = \frac{1}{2}(\frac{1}{2}MR^2)(\frac{v}{R})^2 = \frac{1}{4}Mv^2.$$

Therefore, the ratio of the rotational kinetic energy to the total kinetic energy is

$$\frac{K_r}{K_t + K_r} = \frac{\frac{1}{4}Mv^2}{\frac{1}{2}Mv^2 + \frac{1}{4}Mv^2} = \frac{\frac{1}{4}Mv^2}{\frac{3}{4}Mv^2} = \frac{1}{3}.$$

Final Answer: The final answer is (B).

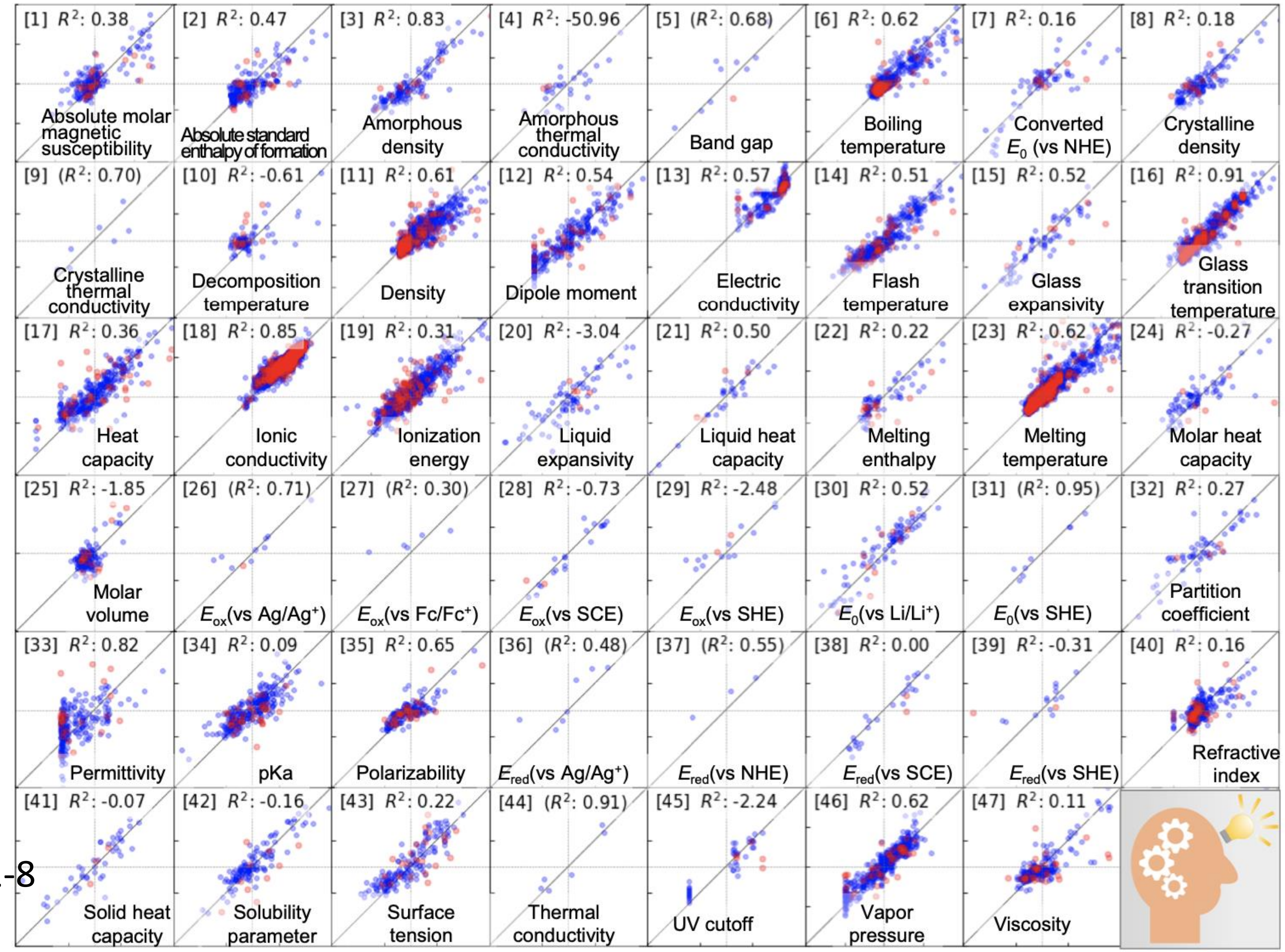
AI for Advanced Property Prediction and Inverse Design

- APP: Predicting a phenotype from genotype (e.g. AMR), Predicting materials properties (e.g. hardness, stability) from composition and symmetry group. Tumor drug response, etc. etc. Protein folding.
- ID: Generate a genome that results in a given phenotype, materials composition and processing steps that results in material given bandgap. Generate a drug that binds to a specific receptor. etc. Design a protein that has a given function.
- Issues: Scale of data DL require O(10^5-10^6)+ examples, Representation of samples/objects, modality, generality, OOD, *learning across problems, general models for multiple systems*
- *Can we develop a model that can do forward and backwards problems across many problem domains?*

Integrating multiple materials science projects in a single neural network

Kan Hatakeyama-Sato [1] & Kenichi Oyaizu[1]✉

COMMUNICATIONS MATERIALS |
https://doi.org/10.1038/s43246-020-00052-8



Predict >40 properties using a single model

Fig. 5 Prediction of versatile parameters. Experimental (*x*-axis) and predicted (*y*-axis) values of each parameter after multitask learning of the 14 databases (90% of data was used for training). Blue and red plots correspond to the training and test datasets, respectively. The values are shown as *z*-scores. R^2 scores for test datasets are displayed. If test cases were not enough for evaluation, train scores are shown with brackets instead. Statistical results are also summarized in Supplementary Table 1. Larger graphs are shown in Supplementary Fig. 14a.

ARTICLE

Check for updates

https://doi.org/10.1038/s41467-022-30839-x

OPEN

Language models can learn complex molecular distributions

Daniel Flam-Shepherd [1,2]✉, Kevin Zhu [1] & Alán Aspuru-Guzik [1,2,3,4]✉

Deep generative models of molecules have grown immensely in popularity, trained on relevant datasets, these models are used to search through chemical space. The downstream utility of generative models for the inverse design of novel functional compounds, depends on their ability to learn a training distribution of molecules. The most simple example is a language model that takes the form of a recurrent neural network and generates molecules using a string representation. Since their initial use, subsequent work has shown that language models are very capable, in particular, recent research has demonstrated their utility in the low data regime. In this work, we investigate the capacity of simple language models to learn more complex distributions of molecules. For this purpose, we introduce several challenging generative modeling tasks by compiling larger, more complex distributions of molecules and we evaluate the ability of language models on each task. The results demonstrate that language models are powerful generative models, capable of adeptly learning complex molecular distributions. Language models can accurately generate: distributions of the highest scoring penalized LogP molecules in ZINC15, multi-modal molecular distributions as well as the largest molecules in PubChem. The results highlight the limitations of some of the most popular and recent graph generative models– many of which cannot scale to these molecular distributions.

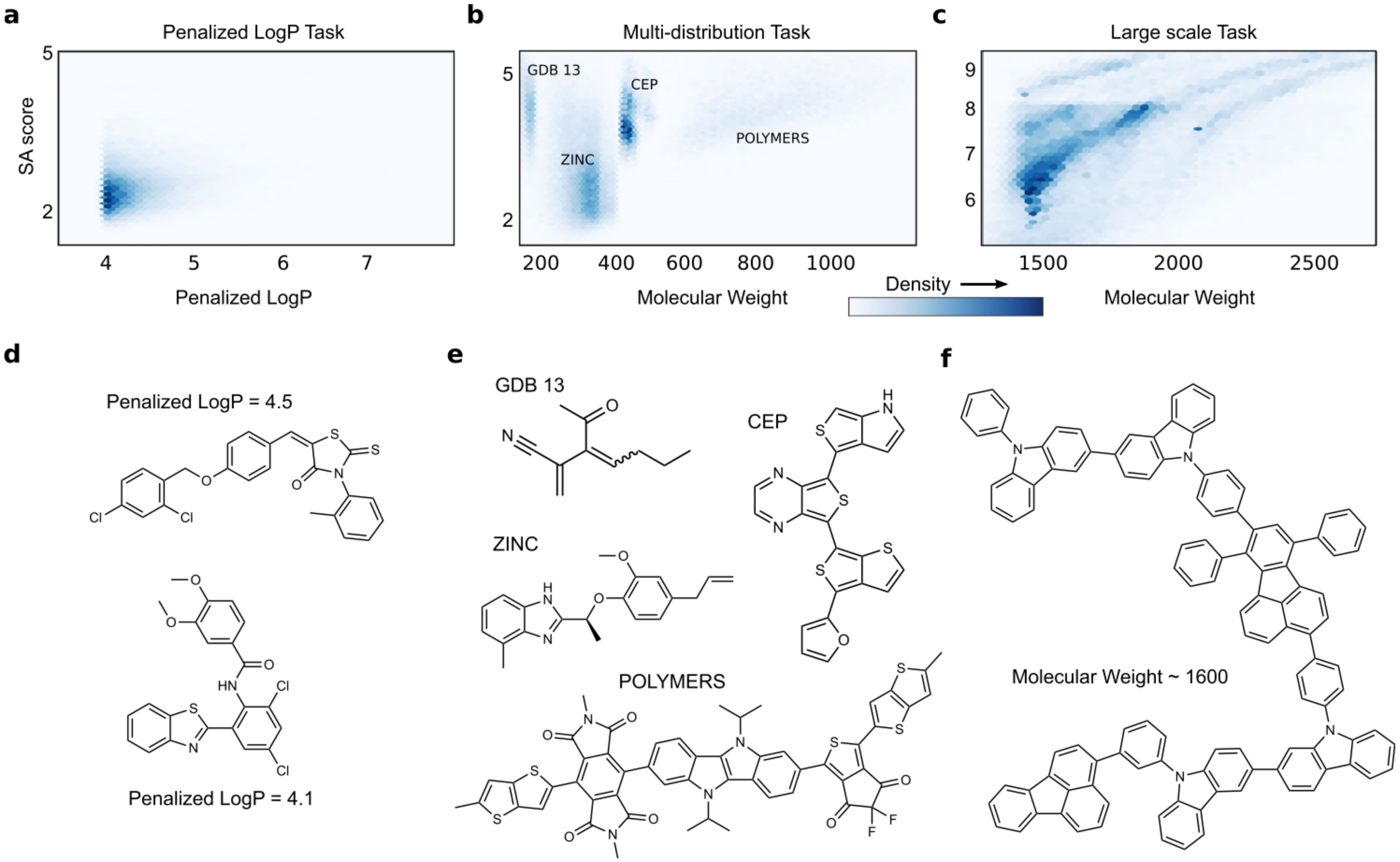
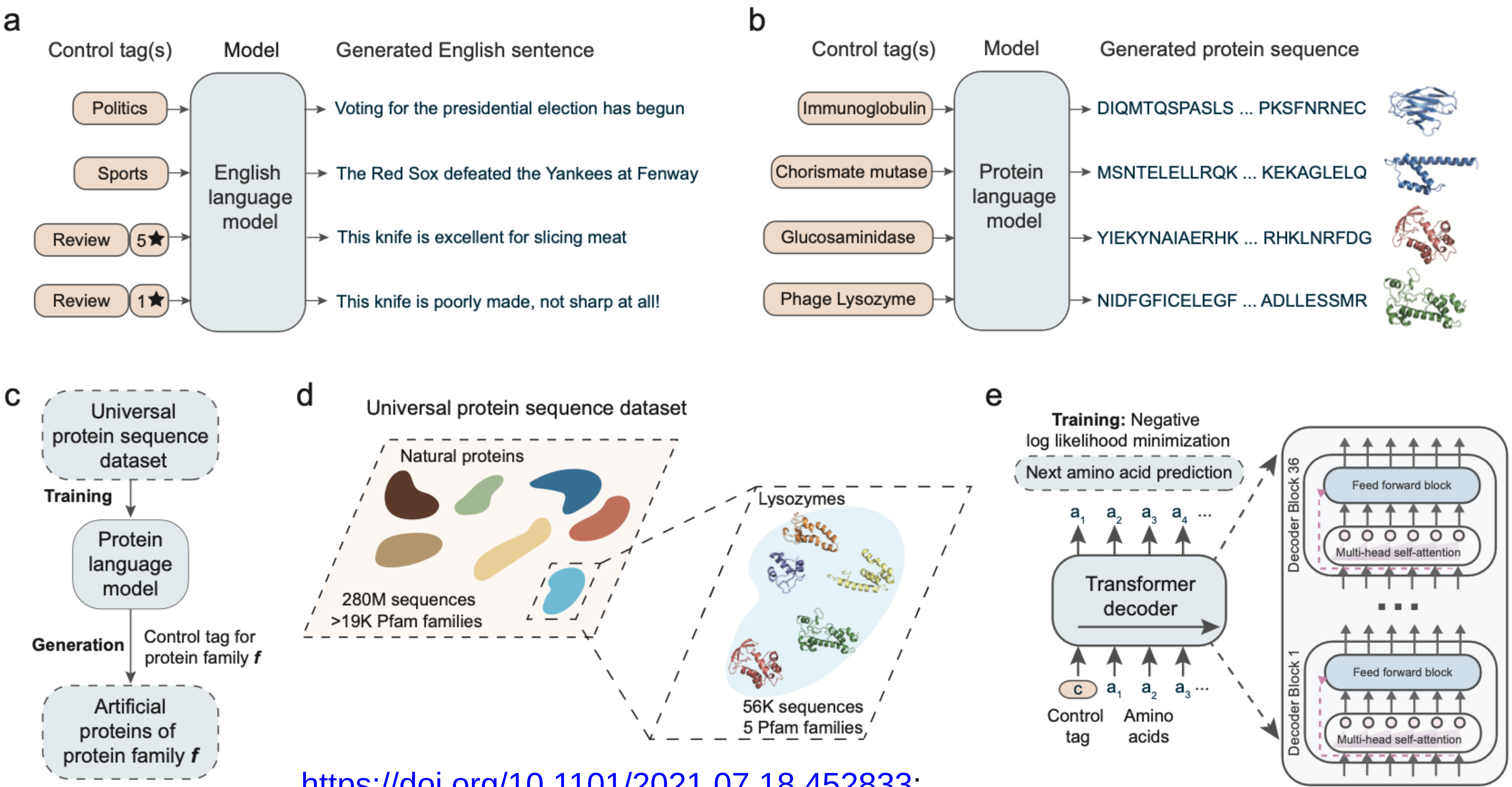


Fig. 1 The generative modeling tasks. a–c The molecular distributions defining the three complex molecular generative modeling task. **a** The distribution of penalized LogP vs. SA score from the training data in the penalized logP task. **b** The four modes of differently weighted molecules in the training data of the multi-distribution task.**c** Large scale task's molecular weight training distribution. **d–f** examples of molecules from the training data in each of the generative modeling tasks. **d** The penalized LogP task, **e** The multi-distribution task. **f** The large-scale task.

Transformers can Design Proteins



https://doi.org/10.1101/2021.07.18.452833;

Artificial protein sequences are functional while reaching as low as 44% identity to any known protein, exhibit comparable catalytic efficiencies to a highly-evolved natural protein, and demonstrate similar structures to known natural folds.

https://doi.org/10.1101/2021.07.18.452833;

AI Based Surrogates for HPC

- Replacing or augmenting computational kernels in HPC applications with machine learned functions that compute that same function but much faster
- Has been demonstrated in many problem domains, physics, climate, CFD, molecular dynamics, drug docking, chemistry, DFT, etc. DeepDrive MD is a good example of a problem that benefits from augmentation.
- Speedups easily > 1000x, sometimes > 100Mx
- Continuous problems mostly, challenges for discrete problems such as those in computational biology (e.g. MSA, sequence similarity, phylogeny)
- Recent progress in replacing hybrid AI/HPC with pure AI, such as Meta's version of AlphaFold that is end-to-end AI, similar accuracy and much faster.
- Open issue is general methods for accelerating discrete problems. Can leverage AI approaches to accelerate sorting and DB lookups, LSH, etc.
- *Differentiable biology programming is a direction for this area, can we systematically replace traditional bioinformatics analysis with methods that can learn from data and be accelerated by AI*

An $O(N)$ Sorting Algorithm: Machine Learning Sort

Hanqing Zhao[1,2] and Yuehan Luo[3]

[1] Department of Modern Physics, University of Science and Technology of China, Hefei 230026, China
[2] School of Physical Science and Technology, and Key Laboratory for Magnetism and Magnetic Materials of MOE, Lanzhou University, Lanzhou, Gansu 730000, China*
[3] Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, Stony Brook, NY, 11794, USA

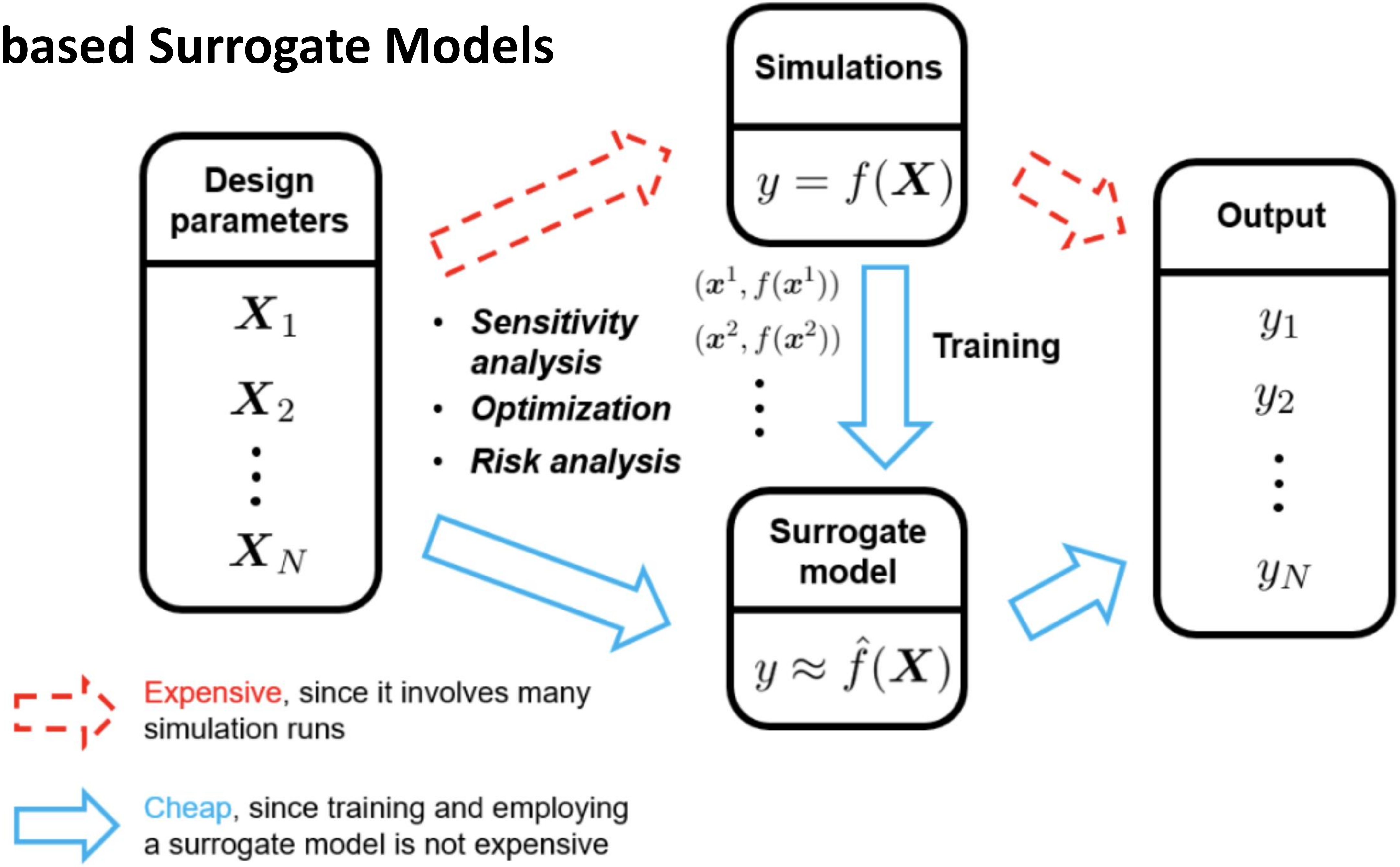
We propose an $O(N \cdot M)$ sorting algorithm by Machine Learning method, which shows a huge potential sorting big data. This sorting algorithm can be applied to parallel sorting and is suitable for GPU or TPU acceleration. Furthermore, we discuss the application of this algorithm to sparse hash table.

INTRODUCTION

Sorting, as a fundamental operation on data, has attracted intensive interests from the beginning of computing [1]. Lots of classic algorithms have been designed and applied, such as Bubble Sort, Selection Sort, Insertion Sort, etc. However, it's been proven that sorting algorithms based on comparison have a fundamental requirement of $\Omega(N \log N)$ comparisons[2, 3], which implies the time complexity is at least $O(N \log N)$[4–6]. The non-comparison sorting algorithms, such as Bucket

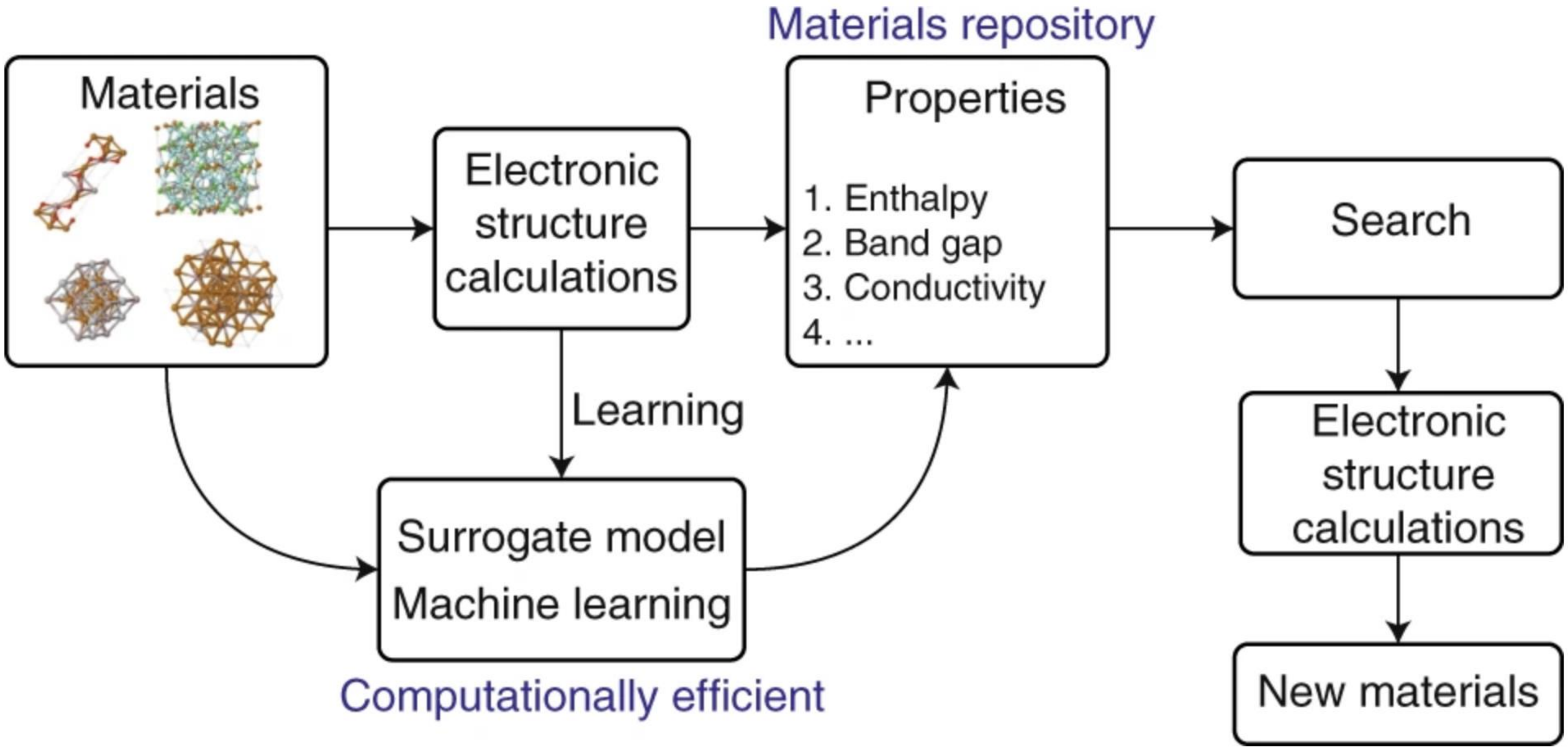
huge success of machine learning shows that computer AI could go beyond human knowledge in complicated tasks, even starting from scratch. After that, machine learning algorithms had been widely applied to various areas such as human vision, natural language understanding, medical image processing, etc., and had achieved great accomplishments. With the breakthrough of these algorithms, the improvements in hardware support the AI algorithms works more efficient, such as GPU/TPU acceleration.

ML based Surrogate Models



As such, the computational efficiency of the LSTM model yields results 42,666 times faster than the full-scale phase-field method.

From: Machine-learned multi-system surrogate models for materials prediction



The accelerated high-throughput approach. Candidate structures and properties are generated by surrogate machine-learning models based on reference electronic structure calculations in a materials repository. Selected structures are validated by electronic structure calculations, preventing false positive errors

MNRAS **000**, 1–6 (2019) Preprint 17 October 2019 Compiled using MNRAS LaTeX style file v3.0

Accelerated simulations up to 100 million times

Newton vs the machine: solving the chaotic three-body problem using deep neural networks

Philip G. Breen[1]*†, Christopher N. Foley[2]*‡, Tjarda Boekholt[3] and Simon Portegies Zwart[4]

[1] *School of Mathematics and Maxwell Institute for Mathematical Sciences, University of Edinburgh, Kings Buildings, Edinburgh, EH9 3JZ*
[2] *MRC Biostatistics Unit, University of Cambridge, Cambridge, CB2 0SR, UK.*
[3] *Instituto de Telecomunicações, Campus Universitário de Santiago, 3810-193, Aveiro, Portugal*
[4] *Leiden Observatory, Leiden University, PO Box 9513, 2300 RA, Leiden, The Netherlands.*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

Since its formulation by Sir Isaac Newton, the problem of solving the equations of motion for three bodies under their own gravitational force has remained practically unsolved. Currently, the solution for a given initialization can only be found by performing laborious iterative calculations that have unpredictable and potentially infinite computational cost, due to the system's chaotic nature. We show that an ensemble of solutions obtained using an arbitrarily precise numerical integrator can be used to train a deep artificial neural network (ANN) that, over a bounded time interval, provides accurate solutions at fixed computational cost and up to 100 million times faster than a state-of-the-art solver. Our results provide evidence that, for computationally challenging regions of phase-space, a trained ANN can replace existing numerical solvers, enabling fast and scalable simulations of many-body systems to shed light on outstanding phenomena such as the formation of black-hole binary systems or the origin of the core collapse in dense star clusters.

Key words: stars: kinematics and dynamics, methods: numerical, statistical

Accelerated simulations up to 2 billion times

Astrophysics
Climate
Biogeochemistry
HEDP
Fusion
Seismology

Building high accuracy emulators for scientific simulations with deep neural architecture search

M. F. Kasim,[1,*] D. Watson-Parris,[2] L. Deaconu,[2] S. Oliver,[3] P. Hatfield,[1] D. H. Froula,[4] G. Gregori,[1] M. Jarvis,[5] S. Khatiwala,[3] J. Korenaga,[6] J. Topp-Mugglestone,[1] E. Viezzer,[7,8] and S. M. Vinko[1]

[1] *Clarendon Laboratory, Department of Physics, University of Oxford, Parks Road, Oxford, UK*
[2] *Atmospheric, Oceanic and Planetary Physics, Department of Physics, University of Oxford, Oxford, UK*
[3] *Department of Earth Sciences, University of Oxford, Oxford, UK*
[4] *Laboratory for Laser Energetics, University of Rochester, New York, USA*
[5] *Denys Wilkinson Building, Department of Physics, University of Oxford, Keble Road, Oxford, UK*
[6] *Department of Geology and Geophysics, Yale University, New Haven, Connecticut, USA*
[7] *Department of Atomic, Molecular and Nuclear Physics, University of Seville, 41012 Seville, Spain*
[8] *Max-Planck-Institut für Plasmaphysik, EURATOM Association, Boltzmannstr. 2, 85748 Garching, Germany*

(Dated: October 9, 2020)

Computer simulations are invaluable tools for scientific discovery. However, accurate simulations are often slow to execute, which limits their applicability to extensive parameter exploration, large-scale data analysis, and uncertainty quantification. A promising route to accelerate simulations by building fast emulators with machine learning requires large training datasets, which can be prohibitively expensive to obtain with slow simulations. Here we present a method based on neural architecture search to build accurate emulators even with a limited number of training data. The method successfully accelerates simulations by up to 2 billion times in 10 scientific cases including astrophysics, climate science, biogeochemistry, high energy density physics, fusion energy, and seismology, using the same super-architecture, algorithm, and hyperparameters. Our approach also inherently provides emulator uncertainty estimation, adding further confidence in their use. We anticipate this work will accelerate research involving expensive simulations, allow more extensive parameters exploration, and enable new, previously unfeasible computational discovery.

I. INTRODUCTION

Finding a general approach to speed up a large class of simulations would enable tasks that are otherwise prohibitively expensive and accelerate scientific research.

limiting their accuracy in emulating simulations with one, two, or three-dimensional output signals. On the other hand, convolutional neural networks (CNN) have shown to have a good prior on natural signals,[8] making them suitable for processing natural n-dimensional

AI for Prediction and Control of Complex Engineered Systems

- Example in this space include controlling fusion reactors, or accelerators, but also can include controlling large-collections of robotics that are carrying out some complex synthesis or experimental protocol.
- Key ideas: Train AI on existing datasets from current systems, predicting outcomes of the controlled system (e.g. plasma or beam stability), then use these models as faster than real-time "digital twin" of the system to predict trajectory of current control and control nudges.
- Key issues: combining a digital twin concept (detailed simulation specific to the device) with AI, possibly an AI surrogate type digital twin to allow exploration of complex system operating phase space.
- Could be applied to engineered biological systems. Cells to Ecosystems. Similar strategy, leverage existing models to generate training data. One model for all taxa?

Opinion piece

Can deep learning beat numerical weather prediction?

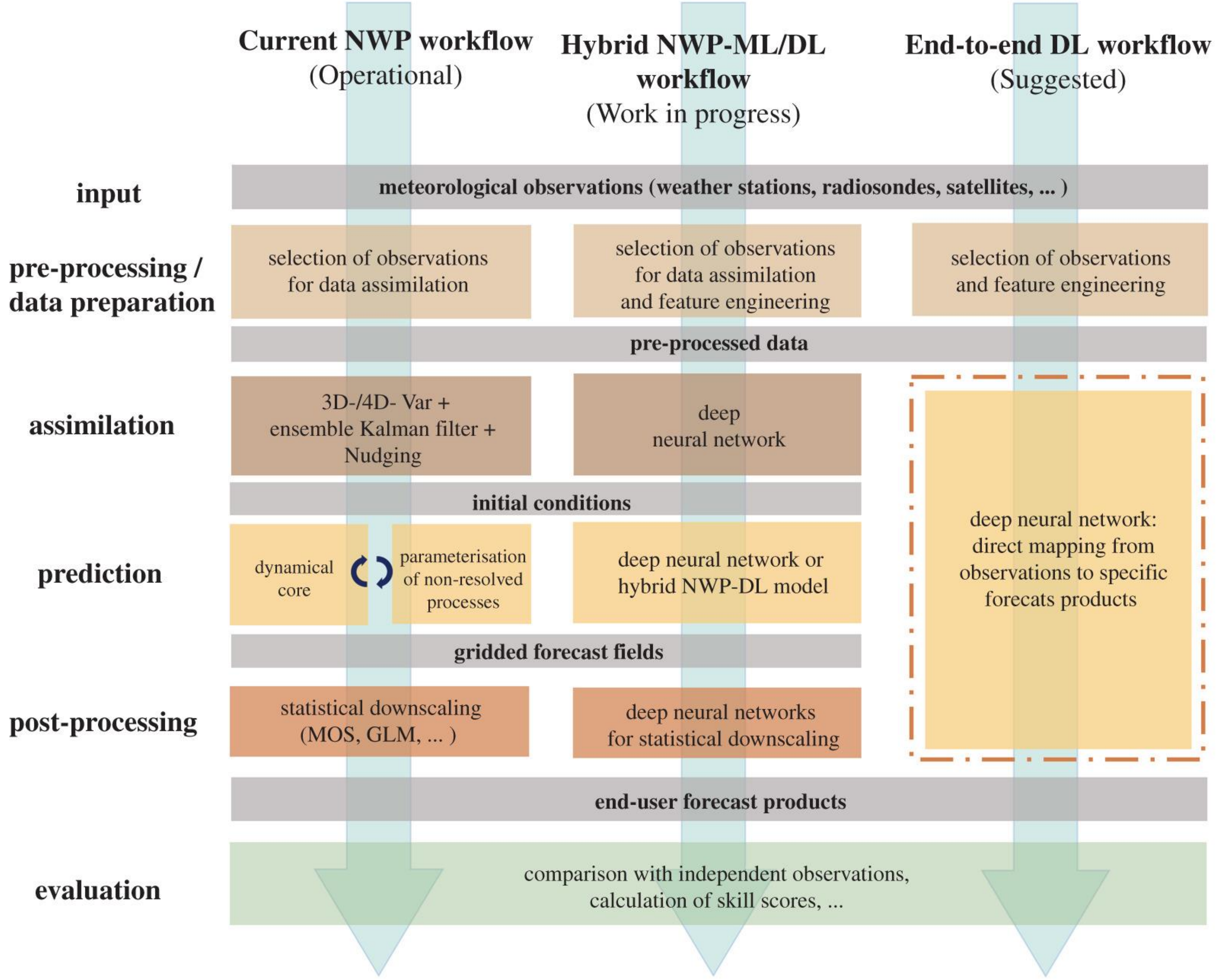
M. G. Schultz, C. Betancourt, B. Gong, F. Kleinert, M. Langguth, L. H. Leufen, A. Mozaffari and S. Stadtler

Published: **15 February 2021** | https://doi.org/10.1098/rsta.2020.0097

Abstract

The recent hype about artificial intelligence has sparked renewed interest in applying the successful deep learning (DL) methods for image recognition, speech recognition, robotics, strategic games and other application areas to the field of meteorology. There is some evidence that better weather forecasts can be produced by introducing big data mining and neural networks into the weather prediction workflow. Here, we discuss the question of whether it is possible to completely replace the current numerical weather models and data assimilation systems with DL approaches. This discussion entails a review of state-of-the-art machine learning concepts and their applicability to weather data with its pertinent statistical properties. We think that it is not inconceivable that numerical weather models may one day become obsolete, but a number of fundamental breakthroughs are needed before this goal comes into reach.

This article is part of the theme issue ‘Machine learning for weather and climate modelling’.


Current NWP workflow
(Operational)
Hybrid NWP-ML/DL workflow
(Work in progress)
End-to-end DL workflow
(Suggested)
input
meteorological observations (weather stations, radiosondes, satellites, ...)
pre-processing / data preparation
selection of observations for data assimilation
selection of observations for data assimilation and feature engineering
selection of observations and feature engineering
pre-processed data
assimilation
3D-/4D- Var + ensemble Kalman filter + Nudging
deep neural network
initial conditions
prediction
dynamical core
parameterisation of non-resolved processes
deep neural network or hybrid NWP-DL model
deep neural network: direct mapping from observations to specific forecats products
gridded forecast fields
post-processing
statistical downscaling (MOS, GLM, ...)
deep neural networks for statistical downscaling
end-user forecast products
evaluation
comparison with independent observations, calculation of skill scores, ...

AI for Programming and Software Engineering

- LLMs are powerful sequence engines, that can be applied to a variety of task in programming and software engineering
  - Code generation from natural language
  - Code completion during editing
  - Translation from one programming lang to another (e.g. Python to C)
  - Debugging code (suggesting bug fixes)
  - Reverse compilation (bin to lang)
  - Automatic Documentation (summarizing code)
  - Code Refactoring for parallelism or new APIs
- Many code task relevant to biology could be targets for AI for programming
- AI for code are typically tuned or augmented language models (e.g. GPT-3, Codex, DeepCoder, etc.)
- AI for programming can also learning to improve automatically using machine generated examples

GPT-3 “DaVinci-002”

Write a simple program for computing a factorial.

One approach to computing a factorial is to use a for loop. We can start with an integer variable n and initialize it to the value we want to compute the factorial of. We can then create a for loop that iterates from 1 to n, and for each iteration we can multiply the loop counter by the current value of n and store the result back in n. After the loop completes, the value of n will be the factorial of the integer we started with.

Here is some pseudocode that implements this approach:

```
n = 5 // we want to compute the factorial of 5

result = 1 // initialize the result to 1

for i = 1 to n

result = result * i // multiply the result by the loop counter

end for

// the value of result is now the factorial of 5
```

Write a simple program for computing a square root.

GPT-3 “DaVinci-002”

The program should take a positive floating point number as input and produce the square root as output.

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
   double n;
   cout << "Enter a positive number: ";
   cin >> n;

   double squareRoot = sqrt(n);
   cout << "Square root of " << n << " = " << squareRoot;

   return 0;
}
```

Write a simple program for computing the roots of a quadratic equation.

GPT-3 “DaVinci-002”

```
#include <iostream>
#include <math.h>

using namespace std;

int main()
{
  float a, b, c, x1, x2, discriminant, realPart, imaginaryPart;
  cout << "Enter coefficients a, b and c: ";
  cin >> a >> b >> c;
  discriminant = b*b - 4*a*c;
  if (discriminant > 0)
  {
    x1 = (-b + sqrt(discriminant)) / (2*a);
    x2 = (-b - sqrt(discriminant)) / (2*a);
    cout << "Roots are real and different." << endl;
    cout << "x1 = " << x1 << endl;
    cout << "x2 = " << x2 << endl;
  }
  else if (discriminant == 0)
  {
    cout << "Roots are real and same." << endl;
    x1 = (-b + sqrt(discriminant)) / (2*a);
    cout << "x1 = x2 =" << x1 << endl;
```

```
  cin >> a >> b >> c;
  discriminant = b*b - 4*a*c;
  if (discriminant > 0)
  {
    x1 = (-b + sqrt(discriminant)) / (2*a);
    x2 = (-b - sqrt(discriminant)) / (2*a);
    cout << "Roots are real and different." << endl;
    cout << "x1 = " << x1 << endl;
    cout << "x2 = " << x2 << endl;
  }
```

```
(pt-light) stevens@M1acBook ~ % g++ quad.cc
(pt-light) stevens@M1acBook ~ % a.out
zsh: command not found: a.out
(pt-light) stevens@M1acBook ~ % ./a.out
Enter coefficients a, b and c: 2 3 4
Roots are complex and different.
x1 = -0.75+1.19896i
x2 = -0.75-1.19896i
```

```
    imaginaryPart =sqrt(-discriminant)/(2*a);
    cout << "Roots are complex and different."  << endl;
    cout << "x1 = " << realPart << "+" << imaginaryPart << "i" << endl;
    cout << "x2 = " << realPart << "-" << imaginaryPart << "i" << endl;
  }
  return 0;
}
```

DeepFix problem.. (code fixing example)

```c
#include <stdio.h>
int main() {
    int a[i];
    int i;
    scanf("%d", &n);
    scanf("%d", &a[i]);
    int count = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == a[i])
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

```c
#include <stdio.h>
int main() {
    int a[100], i, n, count = 0;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == n)
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

Figure 13: An example DeepFix problem with the original broken code on the left and the PaLM-Coder 540B model's prediction on the right. The predicted code contains fixes for all of the compilation errors (undeclared variables), as well as other stylistic improvements (declaring variables together) and logic improvements (reading numbers into array **a** in a loop and not using index **i** outside the loop).

arXiv:2204.02311v2 [cs.CL] 7 Apr 2022

DeepFix problem.. (code fixing example)

```c
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);
    int i, num[400];
    for (i = 0; i < 2 * n; i++)
        scanf("%d", &num[i]);
    int min = 400;
    for (i = 0; i < n; i++) {
        for (int j = 0; j < (2 * n - 1); j++) {
            if (num[i] == num[j])
                int t;
            t = (j - i);
            if (t <= min)
                min = t;
            t = 0;
        }
    }
    printf("%d", min);
    return 0;
}
```

```c
#include <stdio.h>
int main() {
    int n, i, min = 400;
    int num[400];
    scanf("%d", &n);
    for (i = 0; i < 2 * n; i++)
        scanf("%d", &num[i]);
    for (i = 0; i < n; i++) {
        for (int j = 0; j < (2 * n - 1); j++) {
            if (num[i] == num[j]) {
                int t = (j - i);
                if (t <= min)
                    min = t;
            }
        }
    }
    printf("%d", min);
    return 0;
}
```

Figure 14: Another example DeepFix problem. The predicted code fixes the compilation error (missing braces for the `if` block, causing a scope error for variable `t`) and makes other improvements (declaring variables together and removing the line `t = 0;` which has no effect).

arXiv:2204.02311v2 [cs.CL] 7 Apr 2022

Language Models Can Teach Themselves to Program Better

Patrick Haluptzok
Microsoft Research
haluptzok@live.com

Matthew Bowers*
MIT
mlbowers@mit.edu

Adam Tauman Kalai
Microsoft Research
adam@kal.ai

Abstract

This work shows how one can use large-scale language models (LMs) to synthesize programming problems with verified solutions, in the form of "programming puzzles," which can then in turn be used to fine-tune those same models, improving their performance. This work builds on two recent developments. First, LMs have achieved breakthroughs in non-trivial reasoning and algorithm implementation, generating code that can solve some intermediate-level competitive programming problems [e.g., 6, 15, 7]. However, training code LMs involves curated sets of natural-language problem descriptions and source-code tests and solutions, which are limited in size. Second, a new format of programming challenge called a programming puzzle was introduced, which does not require a natural-language description and is directly specified by a source-code test [18]. In this work we show how generating synthetic programming puzzles and solutions, verified for correctness by a Python interpreter, can be used to improve performance in solving test puzzles from P3, a public benchmark set of Python Programming Puzzles. Additionally, we release a dataset of 1 million puzzles and solutions generated by

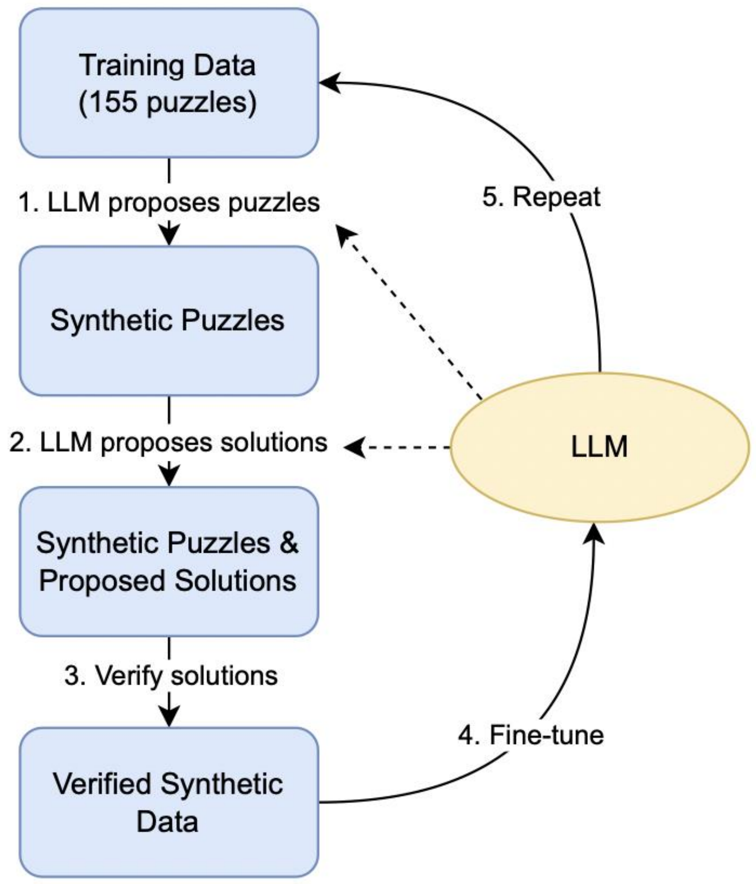
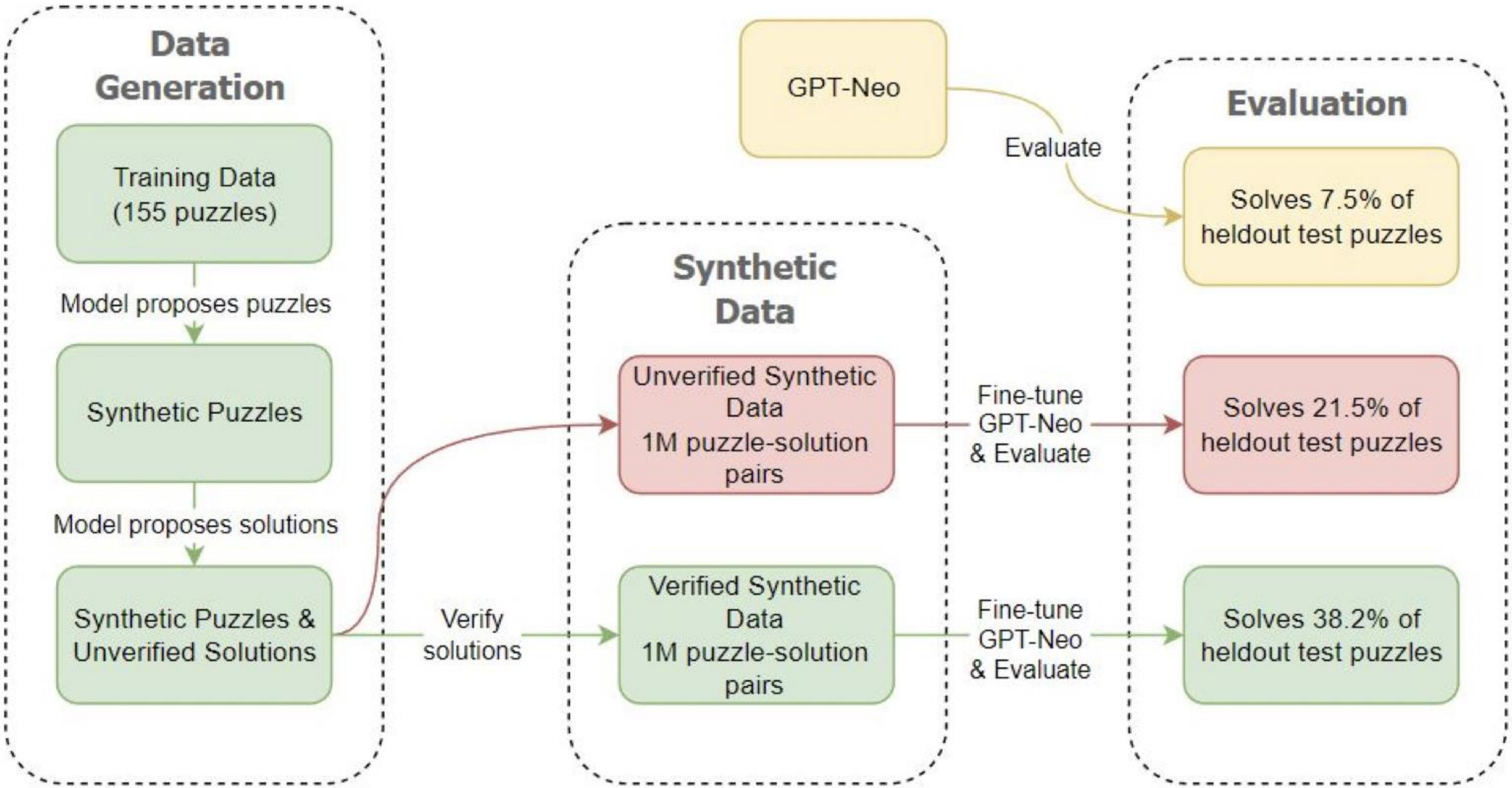
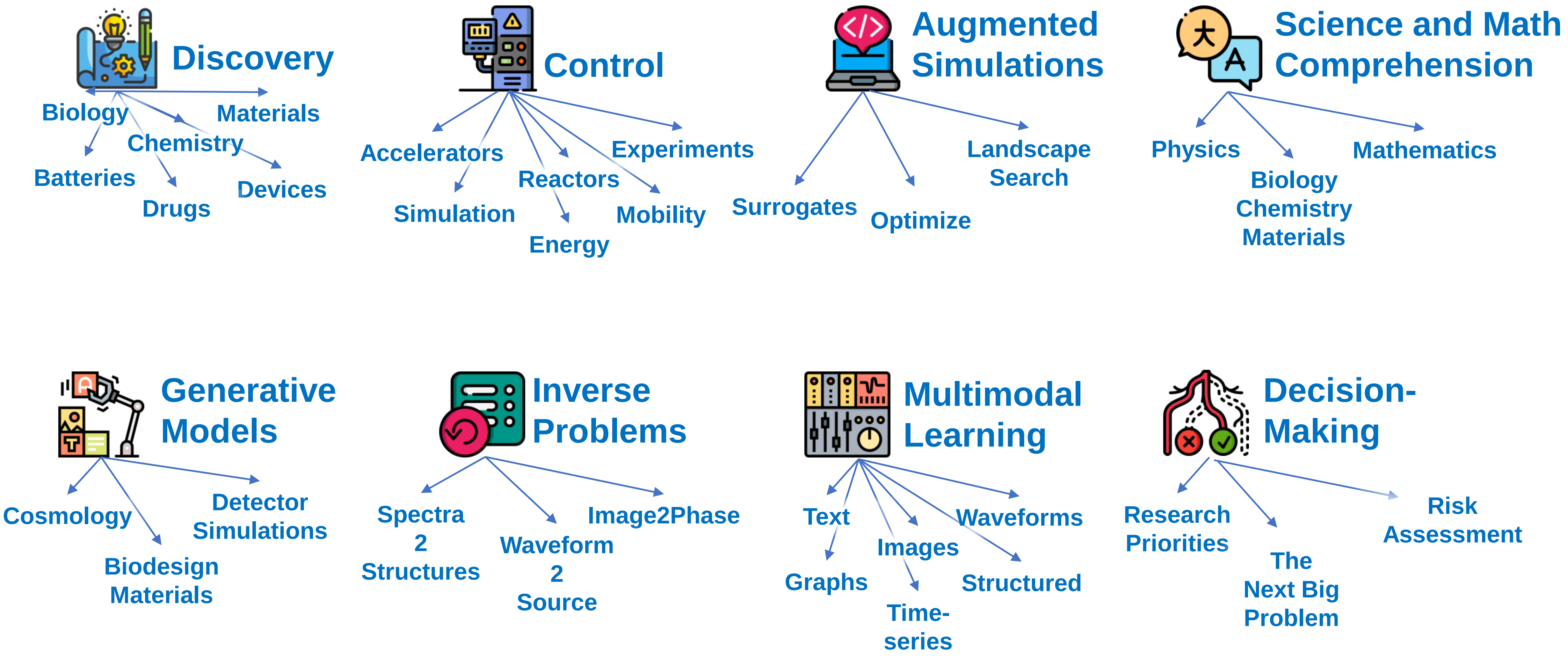


Figure 2: Data Generation Pipeline, used to iteratively generate data and fine-tune the LMs. The pipeline was run on 4 different large language models: Codex and GPT-Neo 125M, 1.3B and 2.7B. Since the API access to Codex doesn't allow for fine-tuning, Codex stopped at step 3 in the first loop after generating 1 million verified puzzles and solutions. GPT-Neo 125M, 1.3B and 2.7B support fine-tuning and we passed through the loop 2 times for the GPT-Neo models. The first loop produced 25K unique puzzle/solution samples from each model to finetune on, a bootstrapping step which greatly sped up the data generation rate in the second loop using the finetuned models. In the second loop we generated 1 million unique puzzle/solution samples from each model to finetune on.


Data Generation
Training Data (155 puzzles)
Model proposes puzzles
Synthetic Puzzles
Model proposes solutions
Synthetic Puzzles & Unverified Solutions
Verify solutions
Synthetic Data
Unverified Synthetic Data 1M puzzle-solution pairs
Verified Synthetic Data 1M puzzle-solution pairs
GPT-Neo
Evaluate
Fine-tune GPT-Neo & Evaluate
Fine-tune GPT-Neo & Evaluate
Evaluation
Solves 7.5% of heldout test puzzles
Solves 21.5% of heldout test puzzles
Solves 38.2% of heldout test puzzles

AI for Science Example Foundation Building Blocks



AI for Science "SuperGeneral" Models

AI Enabled

Design Workflows

(what to make)

→ …materials, polymers, organisms…

AI Enabled

Experimental Workflows

(how to make it)

→ …self-driving labs, synthesis search…

AI Enabled

Scientific Comprehension

(what it means)

→

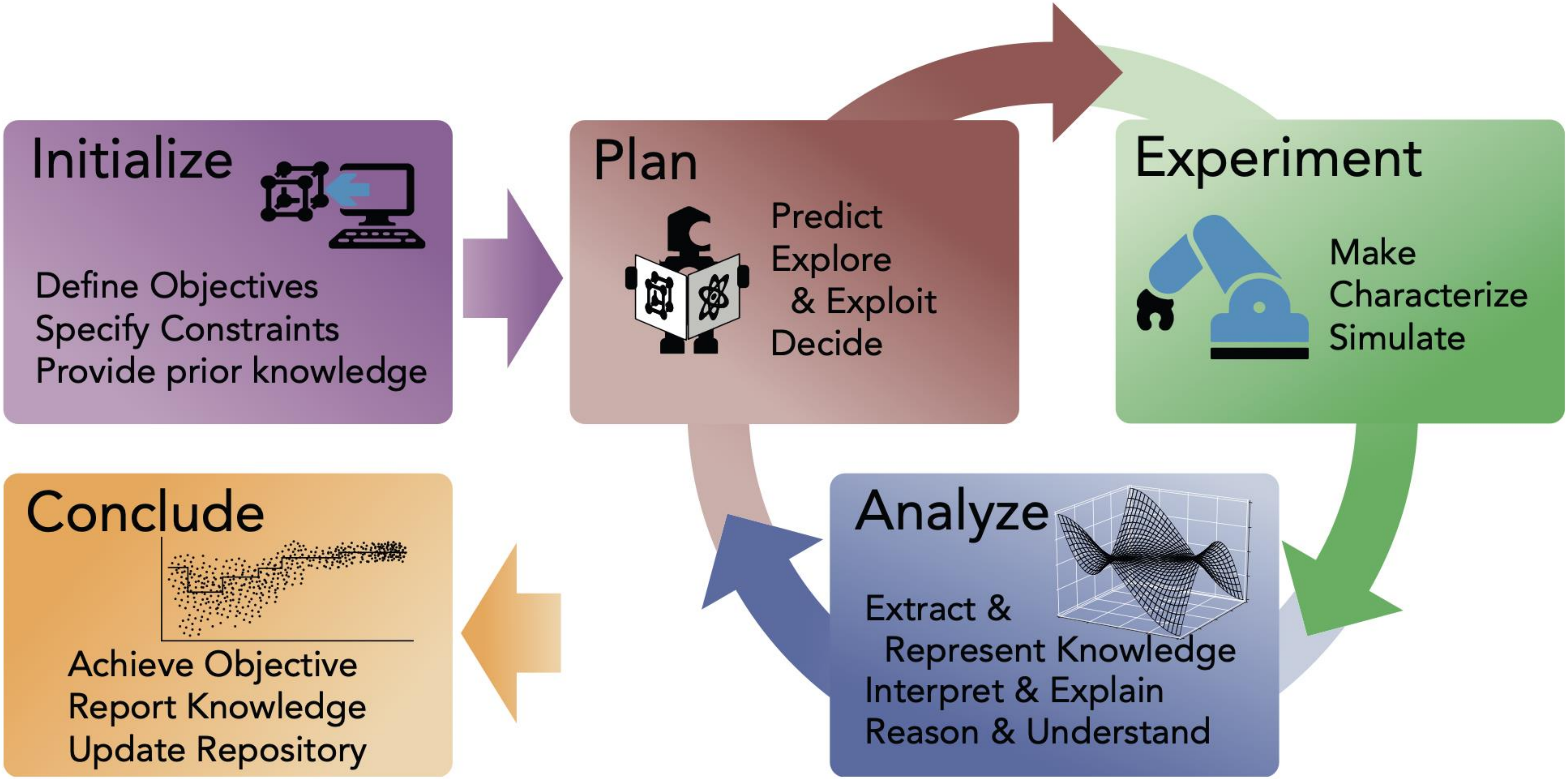
- data Sets
- literature
- science "news"
- strategy

→

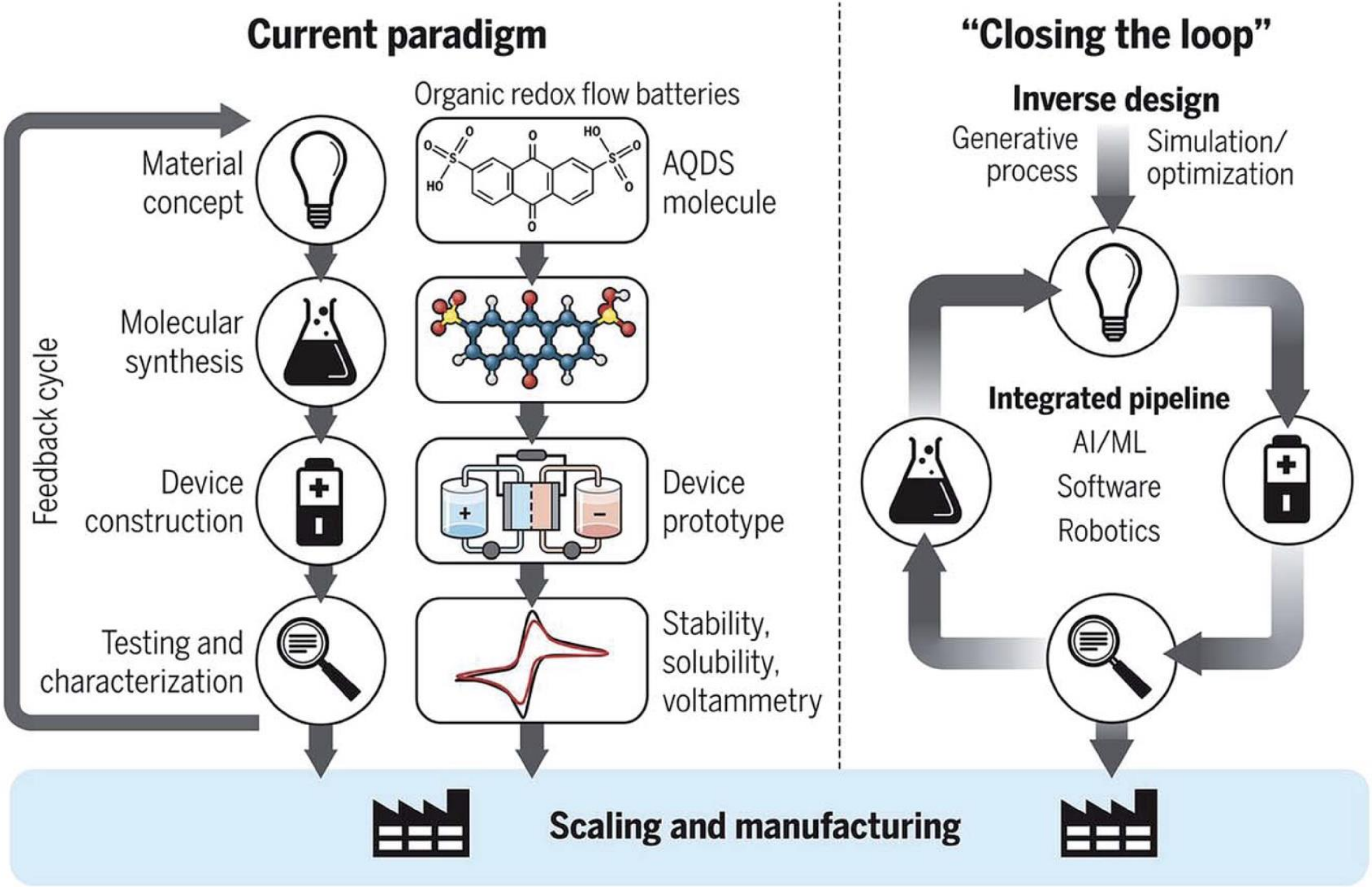
Cleaned
Updated
Annotated
Aggregated
Interpreted

→ Insight?

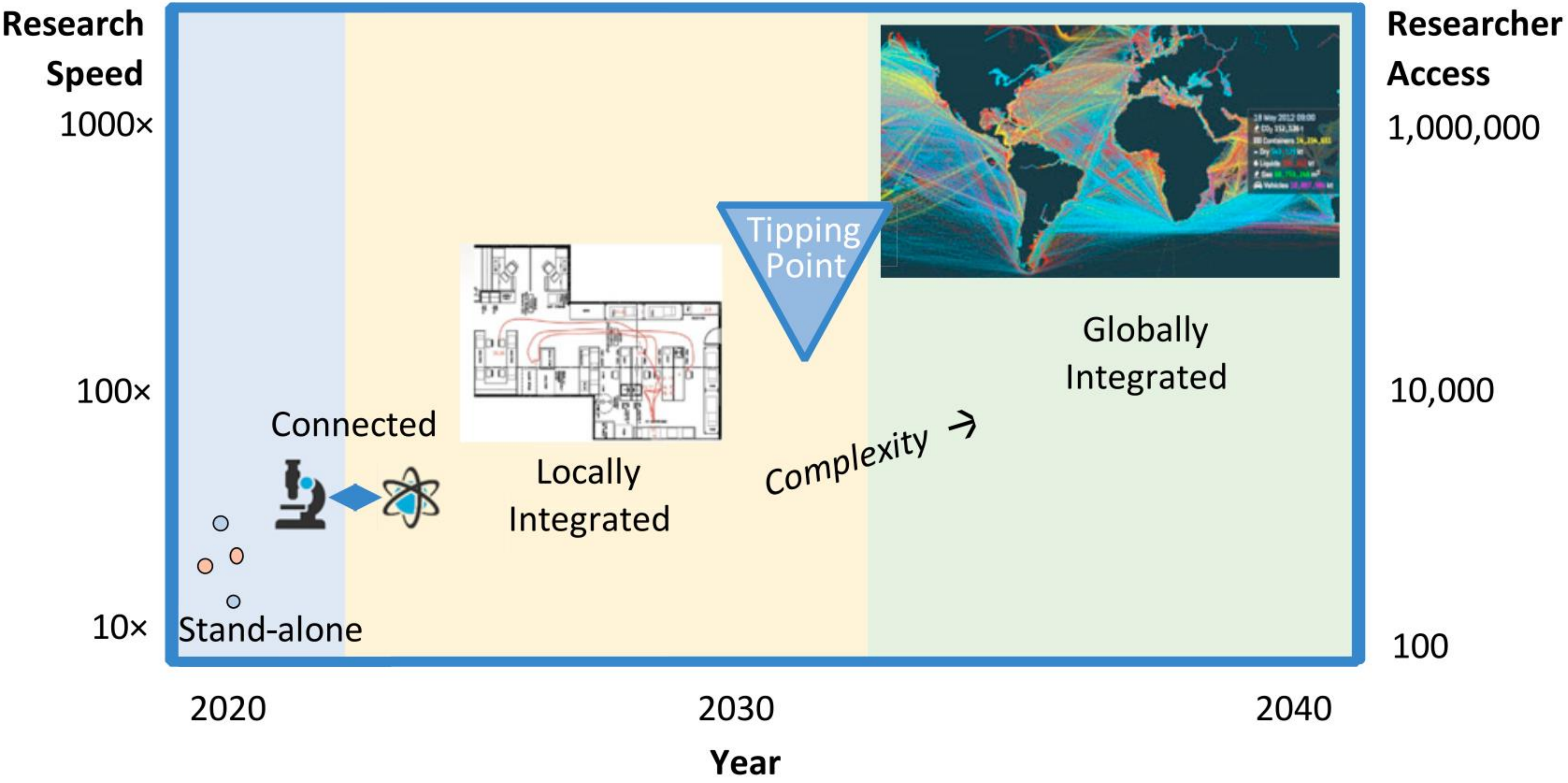
Autonomous Discovery

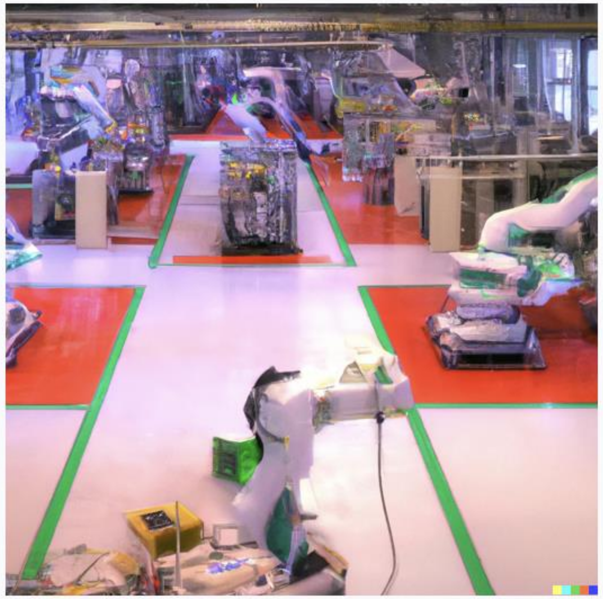


Initialize
Define Objectives
Specify Constraints
Provide prior knowledge
Plan
Predict
Explore
& Exploit
Decide
Experiment
Make
Characterize
Simulate
Analyze
Extract &
Represent Knowledge
Interpret & Explain
Reason & Understand
Conclude
Achieve Objective
Report Knowledge
Update Repository

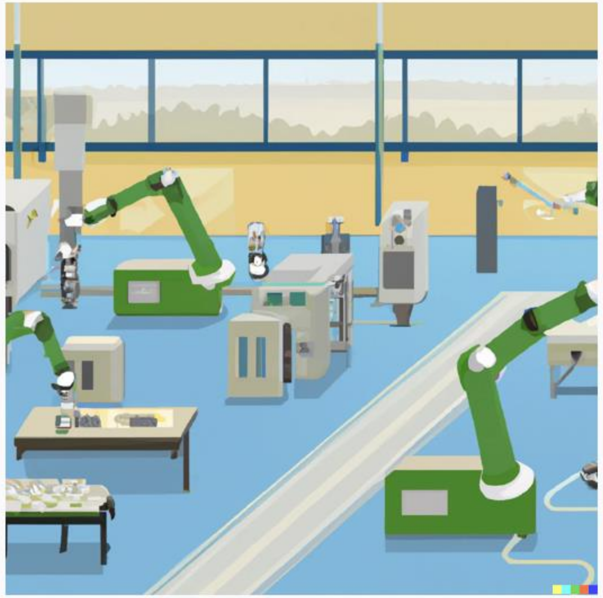
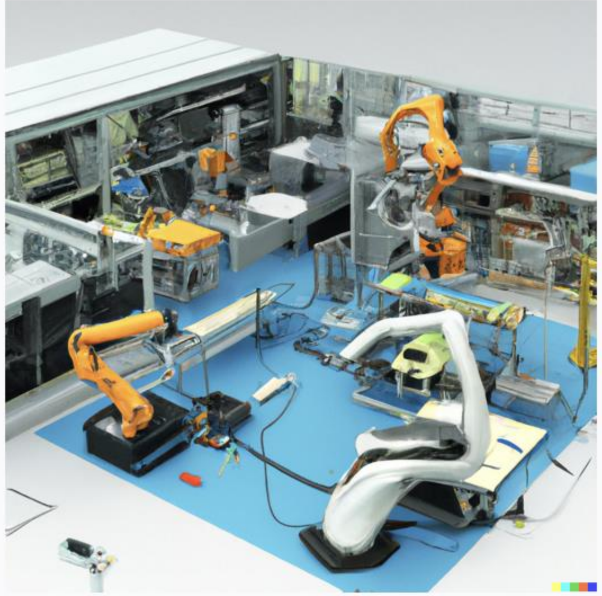
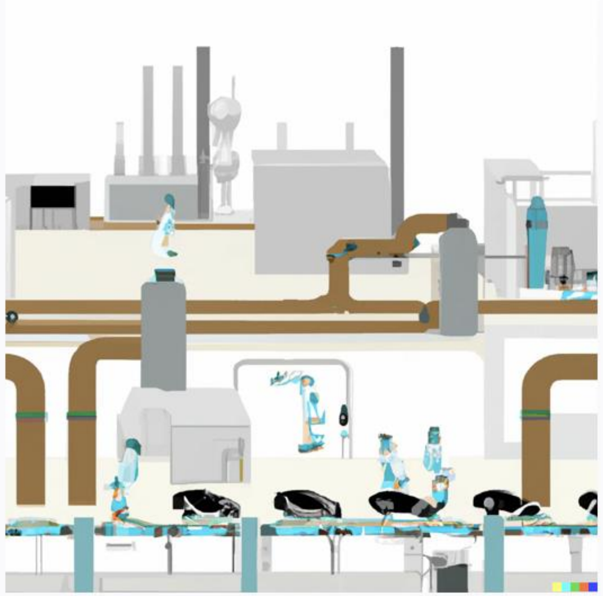
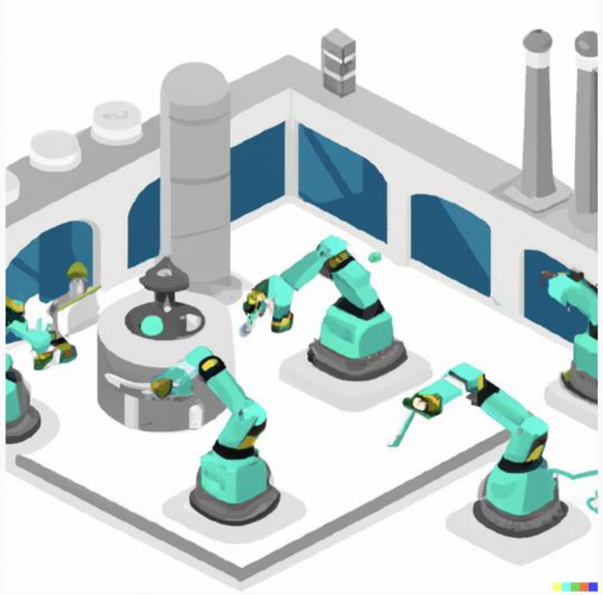

Current paradigm
"Closing the loop"
Organic redox flow batteries
Material concept
AQDS molecule
Molecular synthesis
Feedback cycle
Device construction
Device prototype
Testing and characterization
Stability, solubility, voltammetry
Inverse design
Generative process
Simulation/ optimization
Integrated pipeline
AI/ML
Software
Robotics
Scaling and manufacturing

Expected Progression of Autonomous Experimentation








ORIGINAL