



Why JAX and Flax NNX?

A High-Performance, Flexible Platform for Advanced Computation



Leveraging composable transformations, modern hardware, and a Pythonic neural network API for demanding AI/ML and scientific workloads

Developer Love

“There’s a sense of tranquility when I nuke my code and rewrite it in JAX. Not only does it become faster, all my horrible code is rewritten better”

- Stone Tao (UCSD, Co-Founder of Lux AI Challenge)

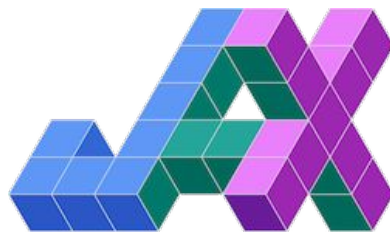
Source: https://twitter.com/Stone_Tao/status/1555394550092353537



Why did Google develop JAX?

- Google learned from years of experience
 - DistBelief
 - TensorFlow
- Google needed high performance to scale efficiently
- Google needed flexibility and modularity to innovate quickly

High performance, flexibility, and modularity became the guiding principles for the development of JAX



Flexibility and Modularity

- You've seen results generated with JAX
- Google uses JAX for nearly all of its research and GenAI development
- Gemini, Gemma, Imagen, Veo, Waymo, etc. are all created using JAX

Article

Highly accurate protein structure prediction with AlphaFold

<https://doi.org/10.1038/s41586-021-03819-2>

Received: 11 May 2021

Accepted: 12 July 2021

Published online: 15 July 2021

Open access

Check for updates

John Jumper^{1,2,3,4}, Richard Evans^{1,2}, Alexander Pritzel^{1,2}, Tim Green^{1,2}, Olaf Ronneberg^{1,2}, Kathryn Tunyasuvunakool^{1,2}, Russ Bates^{1,2}, Aug Anna Potapenko^{1,2}, Alex Bridgland^{1,2}, Clemens Meyer^{1,2}, Simon A. A. Andrew¹, J. Dalgaard¹, Andrew Cowie¹, Bernardino Romero Paredes¹, Rishub Jain¹, Jonas Adler¹, Trevor Back¹, Slig Petersen¹, David Reim¹, Michael Zielinski¹, Martin Steinegger^{1,2}, Michalina Pacholka¹, Tomasz Sebastian Bodenstedt¹, David Silver¹, Oriol Vinyals¹, Andrew W. Senior¹, Paulius M. Kohli¹ & Demis Hassabis^{1,2,3,4}

Proteins are essential to life, and understanding their structure is a key to understanding their function. Through an effort of 4, the structures of around 100,000 unique proteins have been determined, a small fraction of the billions of known protein structures. This has been a long and painstaking task, and the progress has been slow. We have developed a deep learning model, AlphaFold, that can predict protein structure from sequence alone. We have validated an entirely redesigned network-based model, AlphaFold, in the challenging 14th Critical Assessment of Protein Structure Prediction (CASP14), demonstrating accuracy comparable to the best experimental methods. Underpinning the latest version of AlphaFold is a novel approach that incorporates physical and biological knowledge, leveraging multi-sequence alignments, into the design of the model.

Google

PaLM 2 Technical Report

Google*

Abstract

We introduce PaLM 2, a new state-of-the-art language model that has better multilinguality and is more compute-efficient than its predecessor PaLM. PaLM 2 is a Transformer mixture of objectives. Through extensive evaluations on English and multilingual tasks, we demonstrate that PaLM 2 has significantly improved quality on downstream tasks while simultaneously exhibiting faster and more efficient inference compared to PaLM. This enables broader deployment while also allowing the model to respond faster, for a same PaLM 2 demonstrates robust reasoning capabilities exemplified by large improvements on other reasoning tasks. PaLM 2 exhibits stable performance on a suite of responsible inference-time control over toxicity without additional overhead or impact on other capabilities, achieving state-of-the-art performance across a diverse set of tasks and capabilities.

When discussing the PaLM 2 family, it is important to distinguish between pre-training and fine-tuning variants of these models, and the user-facing products that use these models. Products typically include additional pre- and post-processing steps. Additionally, the model is over time. Therefore, one should not expect the performance of user-facing products reported in this report.

Google DeepMind

Gemini: A Family of Highly Capable Multimodal Models

Gemini Team, Google†

This report introduces a new family of multimodal models, Gemini, that exhibit remarkable capabilities across image, audio, video, and text understanding. The Gemini family consists of Ultra, Pro, and Nano sizes, suitable for applications ranging from complex reasoning tasks to on-device memory-constrained use-cases. Evaluation on a broad range of benchmarks shows that our most-capable Gemini Ultra model advances the state of the art in 30 of 32 of these benchmarks — notably being the first model to achieve human-expert performance on the well-studied exam benchmark MMU, and improving the state of the art in every one of the 20 multimodal benchmarks we examined. We believe that the new capabilities of Gemini models in cross-modal reasoning and language understanding will enable a wide variety of use cases and we discuss our approach toward deploying them responsibly to users.

1. Introduction

We present Gemini, a family of highly capable multimodal models developed at Google. We trained Gemini jointly across image, audio, video, and text data for the purpose of building a model with both strong generalist capabilities across modalities alongside cutting-edge understanding and reasoning performance in each respective domain.

Gemini 1.0, our first version, comes in three sizes: Ultra for highly-complex tasks, Pro for enhanced performance and deployability at scale, and Nano for on-device applications. Each size is specifically tailored to address different computational limitations and application requirements. We evaluate the performance of Gemini models on a comprehensive suite of internal and external benchmarks covering a wide range of language, coding, reasoning, and multimodal tasks.

Gemini advances state-of-the-art in large-scale language modeling (Anil et al., 2023; Brown et al., 2020; Chowdhery et al., 2022; Hoffmann et al., 2022; OpenAI, 2023a; Radford et al., 2019; Rae et al., 2021), image understanding (Alayrac et al., 2022; Chen et al., 2022; Dosovitskiy et al., 2020; OpenAI, 2023b; Reed et al., 2022; Yu et al., 2022a), audio processing (Radford et al., 2023; Zhang et al., 2023), and video understanding (Alayrac et al., 2022; Chen et al., 2022). It also builds on the work on sequence models (Sutskever et al., 2014), a long history of work in deep learning based on neural networks (LeCun et al., 2015), and machine learning distributed systems (Barham et al., 2022; Bradbury et al., 2018; Dean et al., 2012) that enable large-scale training.

Our most capable model, Gemini Ultra, achieves new state-of-the-art results in 30 of 32 benchmarks

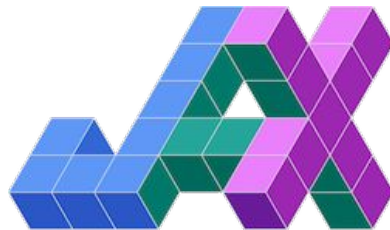
The development of computational methods to predict the three-dimensional (3D) protein structure from the protein sequence has proceeded along two complementary paths that focus on either the physical interactions in the evolutionary history. The physical interaction programme heavily integrates our understanding of molecular driving forces into either thermodynamic or kinetic simulation of protein structure. The second path, however, has been to develop computational methods that focus on the sequence alone, and the rapid development of deep learning methods has led to the emergence of this path. Despite these advances, the current state-of-the-art in protein structure prediction is still far short of the accuracy and reliability of the experimental methods. We have developed a deep learning model, AlphaFold, that can predict protein structure from sequence alone. We have validated an entirely redesigned network-based model, AlphaFold, in the challenging 14th Critical Assessment of Protein Structure Prediction (CASP14), demonstrating accuracy comparable to the best experimental methods. Underpinning the latest version of AlphaFold is a novel approach that incorporates physical and biological knowledge, leveraging multi-sequence alignments, into the design of the model.

We launched Bard as an experiment in March 2023. Since then, we have iterated quickly, always in accordance with our AI Principles. We continue to engage with educators, policymakers, civil rights and human rights leaders, content creators and the many possible applications, as well as the risks and limitations, of this emerging technology. We think Bard is most helpful right now as a standalone experiment. It best allows us

arXiv:2312.11805v1 [cs.CL] 19 Dec 2023

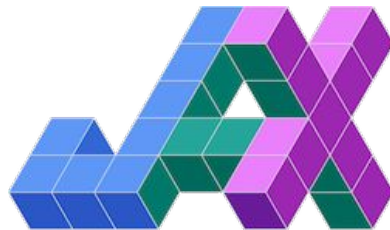
JAX: High-Performance Foundation

- **Accelerated Numerical Computing:** JAX is a platform built on Python and NumPy syntax, designed for high performance on accelerators like GPUs and TPUs.
- **Function Transformations:** Its core power lies in composable function transformations (`jit()`, `grad()`, `vmap()`) that automatically differentiate, compile, and vectorize standard Python code.
- **Beyond ML:** While powerful for ML, JAX is also a foundational library for any domain requiring accelerated numerical operations.



JAX Strength: Scalability & Portability

- **Designed for Scale:** Engineered for speed on single devices and scalability across multiple devices.
- **Simplified Distributed Parallelism:** Leveraging XLA, distributing computations often involves minimal boilerplate; users specify data partitioning (sharding), and XLA handles communication/synchronization.



JAX Strength: Scalability & Portability

- **Automatic Scaling:** Code often scales effectively across different hardware configurations (e.g., single GPU to TPU pods) with minimal changes, as XLA adjusts the execution plan.
- **Hardware Agnosticism:** JAX code typically runs without modification on CPUs, NVIDIA GPUs, and Google TPUs, thanks to XLA abstracting hardware specifics.

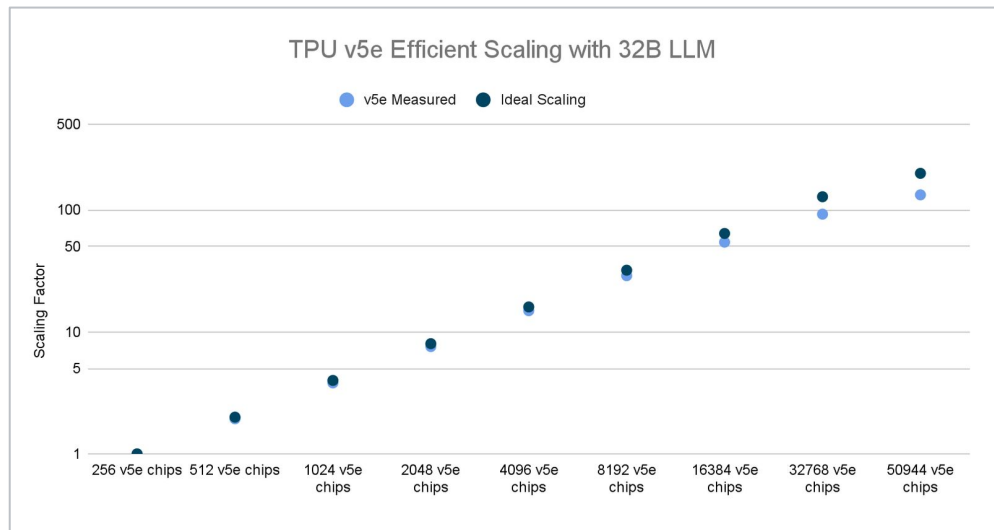


JAX Scalability: Scaling to 50,944 TPUs with JAX

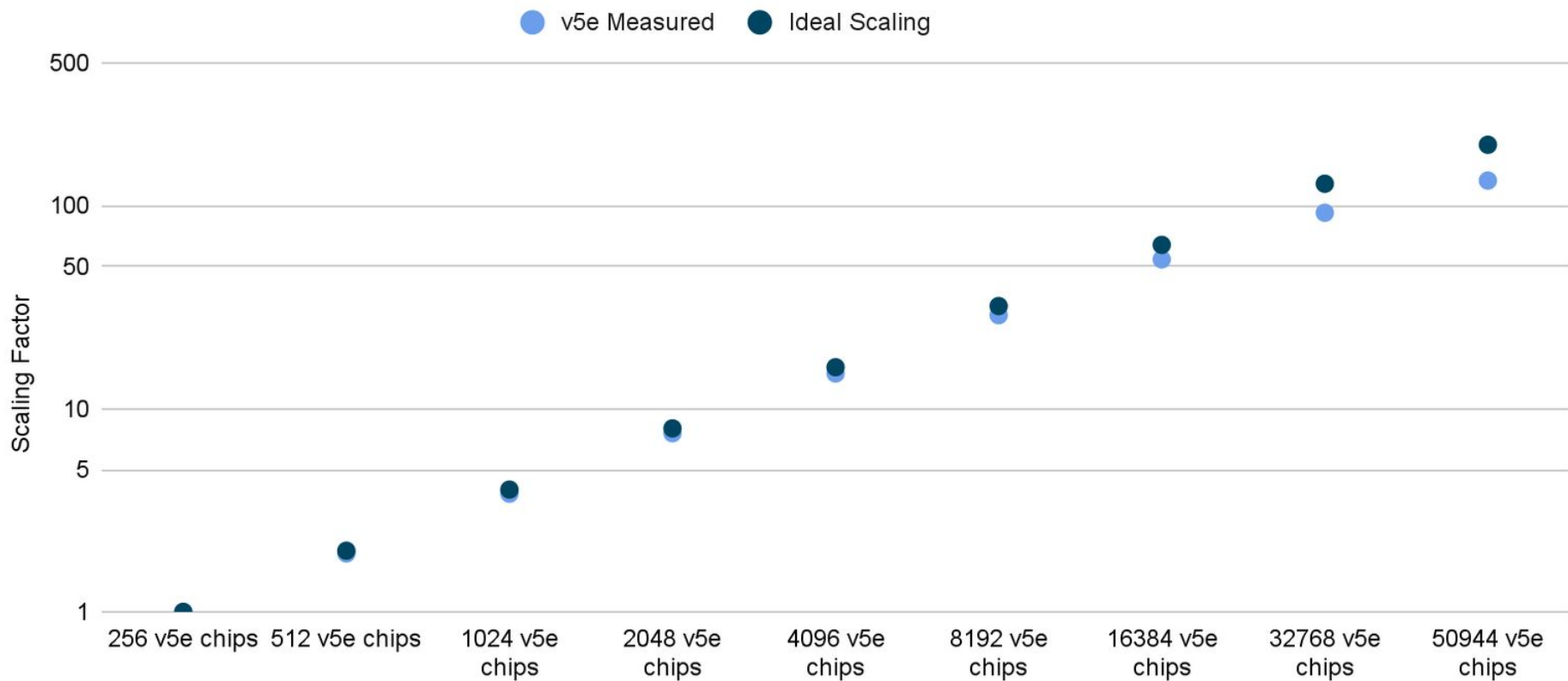
JAX Scalability: TPUs

In November 2023, we used Multislice Training to run an extremely large LLM distributed training job

- 50,944 Cloud TPU v5e chips (spanning 199 Cloud TPU v5e pods)
- Near ideal scaling



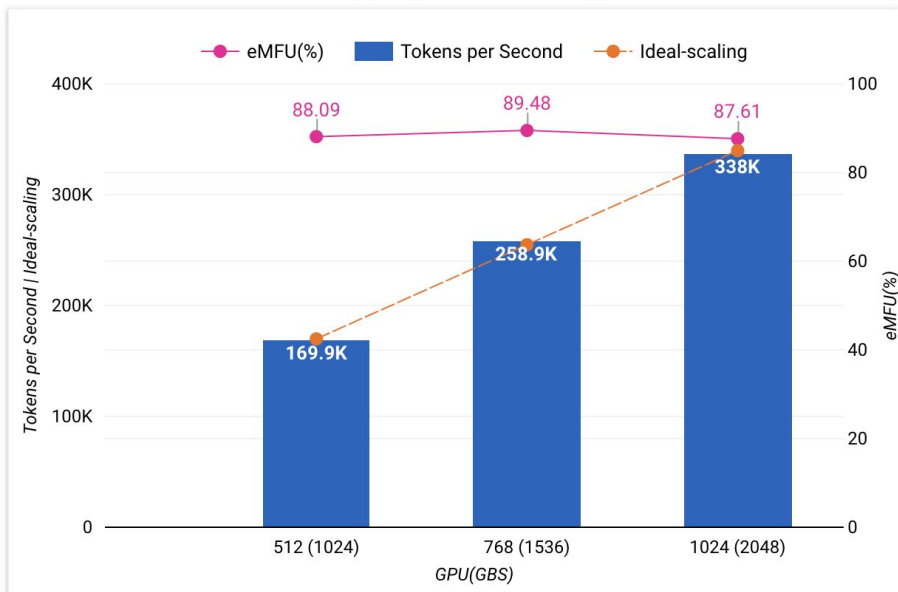
TPU v5e Efficient Scaling with 32B LLM



JAX Scalability: GPUs

Training Performance Results : A3 Ultra

Llama3.1-405B (JAX, MaxText, FP8) Benchmark



High Performance at Scale

> 80% EMFU FP8 at 1000 NVIDIA GPU scale

Near linear scaling upto 1024 GPU scale

MaxText + JAX/XLA stack allows ease of performance optimization

Close partnership with NVIDIA

Reproducible recipes

<https://github.com/Al-Hypercomputer/gpu-recipes/>

*seq-len=8192. emfu= (observed flops throughput/bf16 peak flops). Google internal data for A3 Ultra as of February, 2025.
Performance numbers subject to continuous updates*

Developer Love

“Personally, I've decided to switch to JAX due to its modern approach to parallelism, which can be automatic or semi-automatic. The JAX compiler takes care of many demanding tasks, such as managing the communication of activations and gradients.”

- Luyu Gao (PhD candidate @CarnegieMellon)



Source: https://twitter.com/luyu_gao/status/1768276177448567142

Hardware Portability: JAX v PyTorch v TF Failure Rates

Comparison of TPU and GPU Failure and Success Rates						
	GPUs			TPUs		
	Success	Failure		Success	Failure	
	Pass	Partial	Complete	Pass	Partial	Complete
TensorFlow	78%	8%	14%	71%	15%	14%
PyTorch	92%	3%	5%	57%	27%	17%
JAX	98%	0%	2%	97%	0%	3%

Source:

[The Grand Illusion: The Myth of Software Portability and Implications for ML Progress \(Cohere/MIT Sept 2023\)](#)

Flax NNX

Flax NNX: Intuitive Neural Network Development

Modular, layered design

Flax NNX

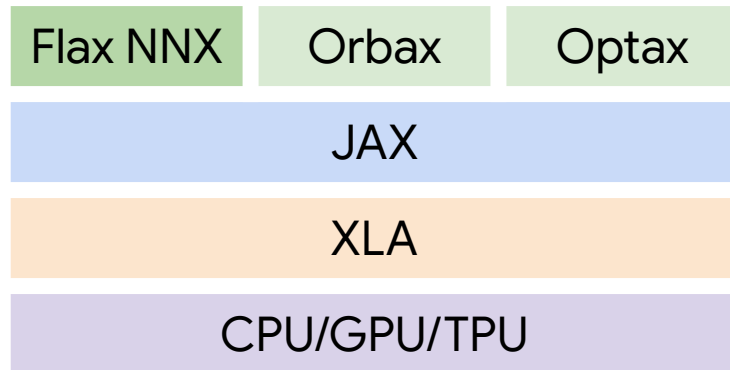
- Neural network library for JAX that is designed for ease of use

Orbax

- Checkpointing and export

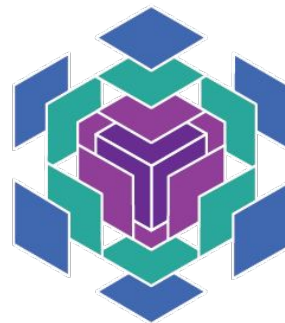
Optax

- Gradient processing and optimization



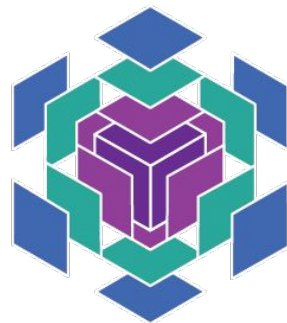
Flax NNX: Intuitive Neural Network Development

- **Modern JAX API:** Flax NNX is a neural network library within the Flax ecosystem, specifically designed for JAX.
- **Simplified & Flexible:** Introduced in 2024, NNX is engineered for simplicity, flexibility, and an enhanced developer experience, making it easier to create, inspect, debug, and analyze models.
- **First-Class Pytree Integration:** NNX Modules are now native JAX Pytrees, allowing direct use with `jax.jit`, `jax.vmap`, and other transformations.
- **Builds on Experience:** Incorporates learnings from previous JAX libraries for a more user-friendly interface.



Flax NNX Strength: Pythonic Design

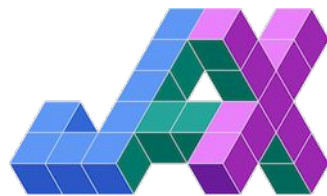
- **Familiar Object Semantics:** Embraces standard Python object concepts like classes, inheritance, attributes, methods, and reference semantics. Allows natural patterns like weight sharing.
- **Intuitive Module Definition:** Define layers/models by subclassing `nnx.Module`, defining sub-layers as attributes in `__init__`, and logic in `__call__`. Uses standard layers like `nnx.Linear`, `nnx.Conv`, `nnx.BatchNorm`, etc.
- **Seamless JAX Integration:** As native Pytrees, NNX modules work directly with JAX's function transformations.
- **Easier Adoption:** Lowers the barrier for developers familiar with object-oriented frameworks (like PyTorch/Keras) to leverage JAX.



JAX Ecosystem

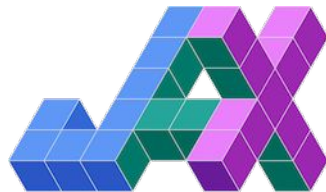
JAX Ecosystem: Breadth & Flexibility

- **Large and Growing:** A major strength is the large, diverse ecosystem of libraries, tools, and projects built upon JAX.
- **Testament to Power:** This vibrant activity showcases JAX's flexibility and applicability across a wide array of domains, far beyond conventional deep learning.
- **Modular Philosophy:** JAX core remains lean, encouraging domain-specific innovation in independent libraries.
- **Curated Neural Network Stack:** The JAX AI Stack provides a tested set of core libraries (JAX, Flax, Optax, Orbx) for compatibility and easier onboarding.



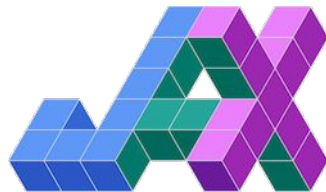
JAX Ecosystem: Representative Examples

- **Neural Networks:** Flax NNX, Equinox, Penzai, Scenic, Objax, EasyDeL.
- **Foundation Models:** MaxText, Levanter, EasyLM, Marin.
- **Reinforcement Learning:** RLax, BRAX (physics), gymnax (envs), Jumanji (envs), Mctx (search), Pgx (games).
- **Probabilistic Programming:** NumPyro, Oryx, Distrax, BlackJAX (samplers), GPJax (Gaussian Processes).



JAX Ecosystem: Representative Examples

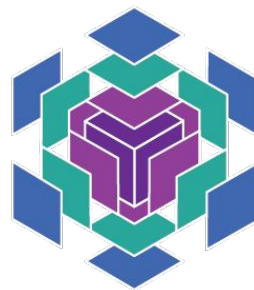
- **Scientific Computing:** JAX M.D. (molecular dynamics), NetKet (quantum physics), jax-cosmo (cosmology), Diffrax (diff eq solvers), delta PV (photovoltaics), dynamiqs (quantum dynamics), XLB (fluid dynamics).
- **Optimization:** Optax, JAXopt, Optimistix.
- **Utilities:** Chex (testing), Orbx (checkpointing), SafeJax (serialization).



Conclusion

Conclusion: Premier Choice for Advanced Computation

- **JAX Foundation:** Provides exceptional performance via XLA, `jit()`, `grad()`, `vmap()`, enhanced by a functional paradigm promoting composability and reproducibility.
- **Flax NNX Usability:** Offers an intuitive, Pythonic API for building, debugging, and managing neural networks, with native Pytree integration that connects seamlessly with JAX.
- **Powerful Combination:** Together, JAX's performance and Flax NNX's usability, amplified by the vast and diverse ecosystem, create a leading platform for tackling complex challenges in AI/ML and scientific discovery.



Learning Resources

Code Exercises, Quick References, and Slides

- <https://goo.gle/learning-jax>

More videos for learning JAX

- <https://goo.gle/learn-jax-videos>



Learn JAX

Community and Docs

Community:

- <https://goo.gle/jax-community>

Docs

- JAX AI Stack: <https://jaxstack.ai>
- JAX: <https://jax.dev>
- Flax NNX: <https://flax.readthedocs.io>