

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 Al 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



Performance: JAX versus NumPy

```
def jax_batch(W, b, x):
  for i in range(10000):
    result = jax_predict(W, b, x)
  return 0
def np_batch(W, b, x):
 for i in range(10000):
    result = np_predict(W, b, x)
  return 0
```

jit_predict = jax.jit(jax_batch)

Colab CPU Instance:

```
NumPy: np_batch(W, b, x)
92.7 ms ± 22.7 ms per loop

JAX (with compile): jit_predict(W, b, x)
34.6 µs ± 41.8 µs per loop

JAX (after compile): jit_predict(W, b, x)
17.3 µs ± 6.83 µs per loop
```

Performance: NNX versus PyTorch

```
def nnx_batch(W, b, x):
    for i in range(10000):
       result = nnx_predict(W, b, x)
    return 0

def pt_batch(W, b, x):
    for i in range(10000):
       result = pt_predict(W, b, x)
    return 0
```

```
Colab CPU Instance:

PyTorch: pt_batch(W, b, x)
82.2 ms ± 18.8 ms per loop

NNX (with compile): nnx_jit_predict(W, b, x)
83.1 µs ± 76.3 µs per loop

NNX (after compile): nnx_jit_predict(W, b, x)
37.2 µs ± 1.42 µs per loop
```

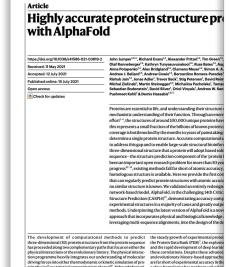
```
nnx_jit_predict = nnx.jit(nnx_batch)
```

Flexibility and Modularity

You've seen results generated with JAX

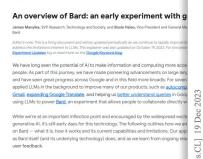
reported in this report.

- Google uses JAX for nearly all of its research and GenAl development
- Gemini, Gemma, Imagen, Veo, Waymo, etc. are all created using JAX





Google



What Bard is

Bard is designed as an interface to an LLM that enables users to collaborate with genone of the promises of LLM-based innovations like Bard is to help people unlock their they can augment their imagination, expand their curiosity, and enhance their produc

We launched Bard as an experiment in March 2023. Since then, we have iterated quick capabilities – always in accordance with our <u>Al Principles</u>. We continue to engage wit educators, policymakers, civil rights and human rights leaders, content creators and of the many possible applications, as well as the risks and limitations, of this emerging to contact the content of the

We think Bard is most helpful right now as a standalone experiment. It best allows us to

Google DeepMind

Gemini: A Family of Highly Capable Multimodal Models

Gemini Team, Google¹

This report introduces a new family of multimodal models, Gemini, that exhibit remarkable capabilities across image, and, 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 loved range of beet/marks 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 benchmarks MuRUL, and improving the state of the art in every one of the 20 multimodal benchmarks we starmined. 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 use.

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 asilored 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 frange 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; Chowdheyer et al., 2023; Horman et al., 2022; OpenAI, 2023; Rafford et al., 2019; Rue et al., 2021), image understanding (Alayrac et al., 2022; Chent et al., 2022; Dossovitskiy et al., 2020; OpenAI, 2023). Freed et al., 2022; Vut et al., 2022a, and to processing (Enderd et al., 2023). Tablo builds on the work on sequence models (Sustkeev et al., 2014). a long history of work in deep learning observed on neural networks (LCCum et al., 2015), and machine learning distributed systems (Barham et al., 2022; Brand et al., 2015) and machine learning distributed systems (Barham et al., 2022; Brand et al., 2015).

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

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.



Developer Love

"I used to write custom CUDA kernels and optimize my code to stuff large GNNs into GPU memory. So I decide to profile some common GNN operations with JAX and PyTorch. It turns out that the JIT of JAX always outperforms PyTorch in both time and mem."



- Zhaocheng Zhu (PhD @ MilaQuebec)

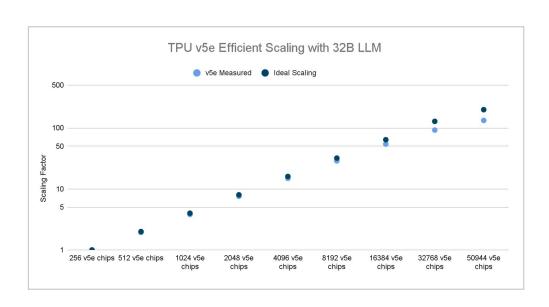
Source: https://twitter.com/zhu_zhaocheng/status/1656372666582827008

JAX Scalability: Scaling to 50,944 TPUs with JAX

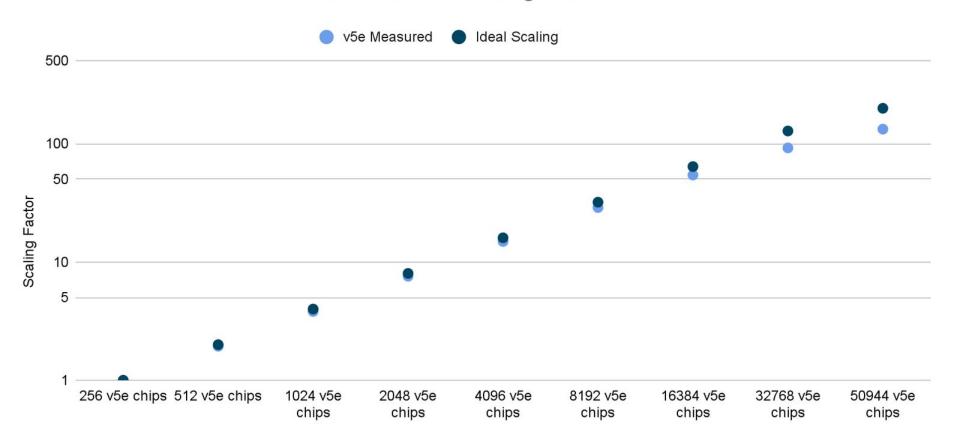
JAX Scalability: TPUs

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

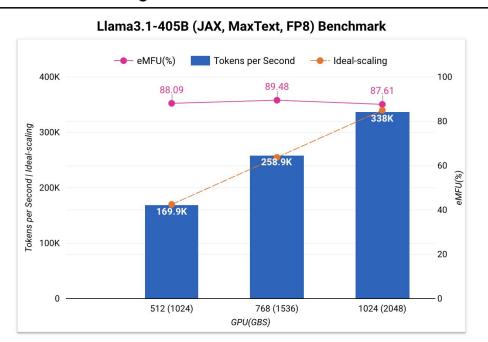


TPU v5e Efficient Scaling with 32B LLM



JAX Scalability: GPUs

Training Performance Results: A3 Ultra



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/

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					
	\mathbf{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

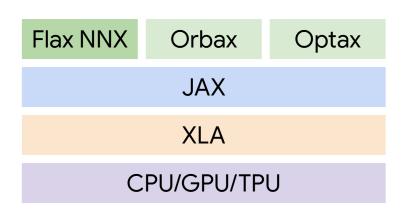
 Neural network library for JAX that is designed for ease of use

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.



Developer Love

"The jit'd JAX (Flax in this case) impl ran circles around familiar PyTorch and TF impl of similar algos (CPU or GPU). And that's without taking the time to vectorize the eval envs. I was using https://github.com/ikostrikov/jaxrl as my starting point."



- Ross Wightman

Source: https://twitter.com/wightmanr/status/1417616247118733313

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, Orbax) 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), Orbax (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



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