NASA-STIG — AD / PyTorch / JAX

Accelerated Computation with Auto-differentiation

Phill Cargile
Research Scientist
Center for Astrophysics | Harvard & Smithsonian
Dec. 15th 2025

Outline:

- Autodiff Motivation: Why do we need anything beyond NumPy & SciPy?
- PyTorch (Dec. 15th)
  - Model Building
  - Training
- JAX + JAX Ecosystem (Dec. 22nd)

NumPy & SciPy

NumPy

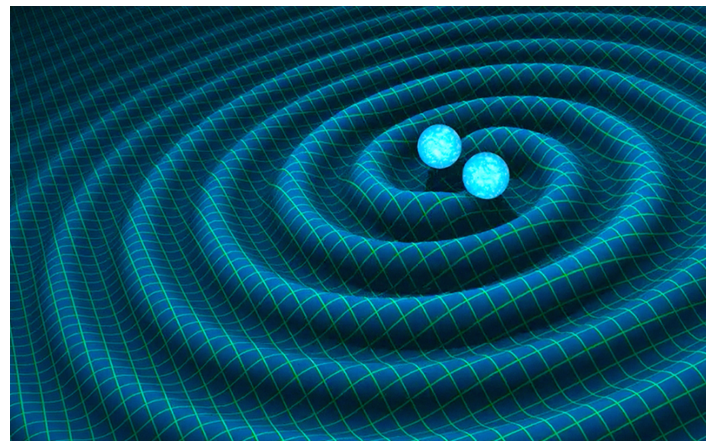
Tasks:

- ◆Efficient N-D Array Math
- ◆Vectorized Operations
- ◆Broadcasting
- ◆Statistical Functions
- ◆Linear Algebra
- ◆FFT
- ◆Random Numbers

SciPy.org

Tasks:

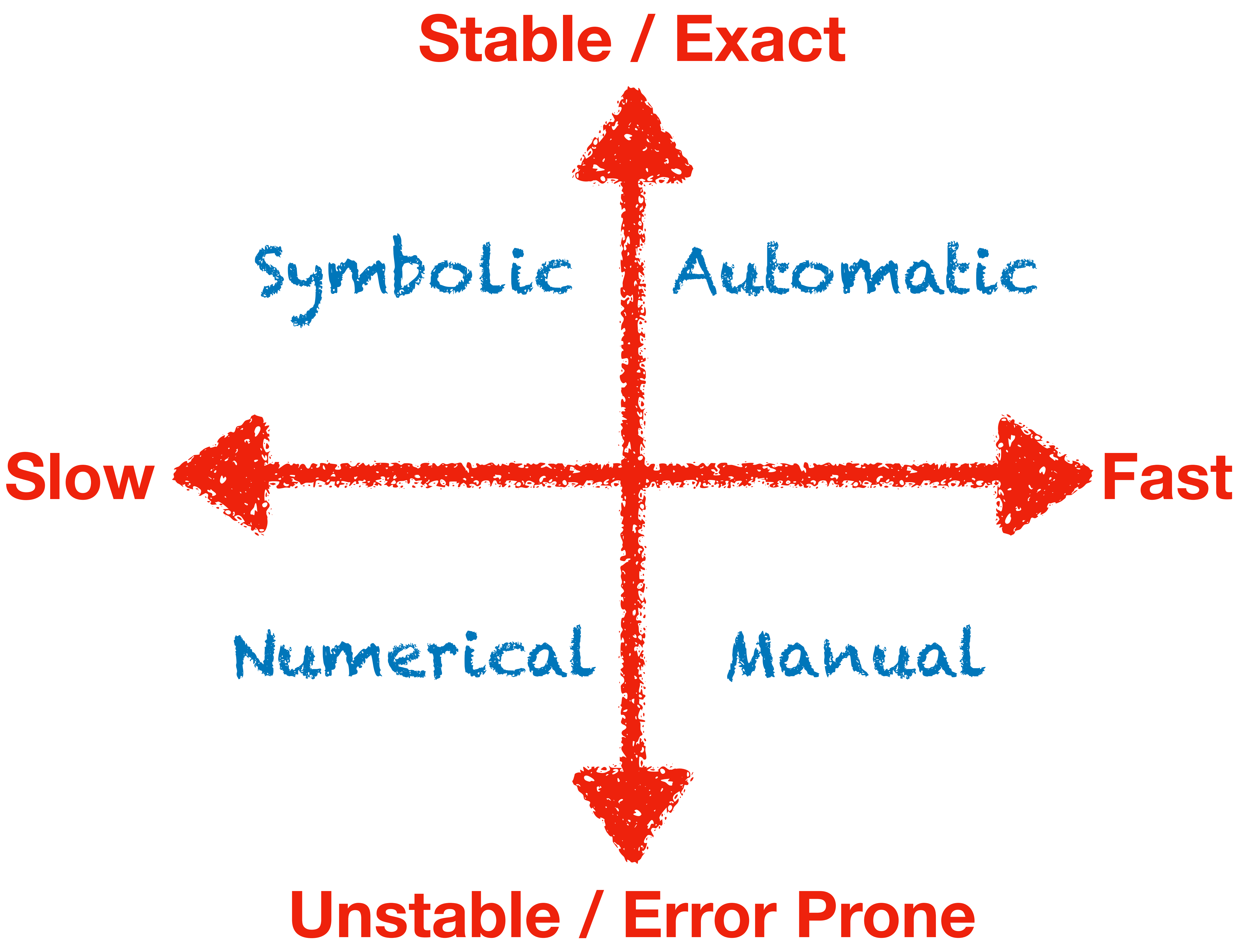
- ◆Regression / Optimization
- ◆Signal Processing
- ◆Interpolation
- ◆Integration



The Astropy Project

What is the partial derivative of a Python function?

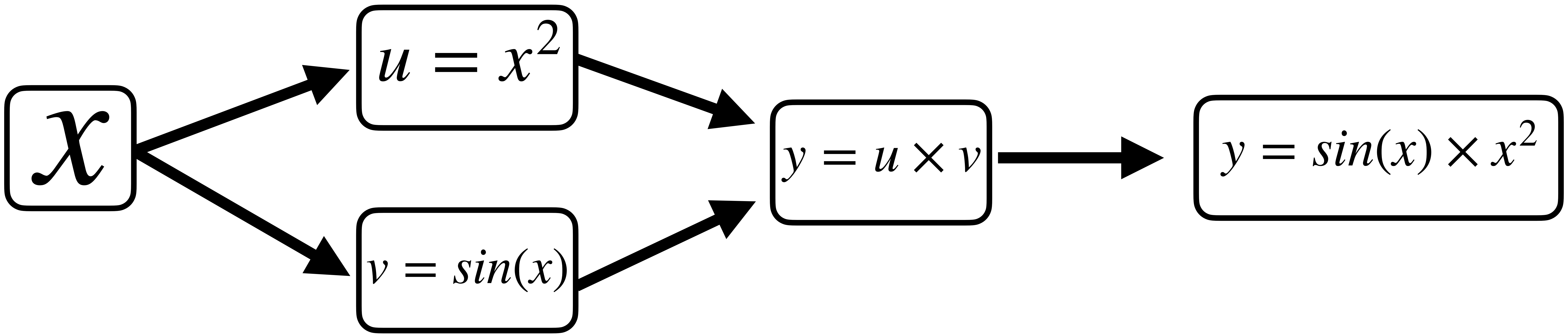
Ways to compute a derivative:



Automatic Differentiation (Autodiff, AD, etc.)

$f(x) \rightarrow y$

$$\boxed{y = sin(x) \times x^2}$$



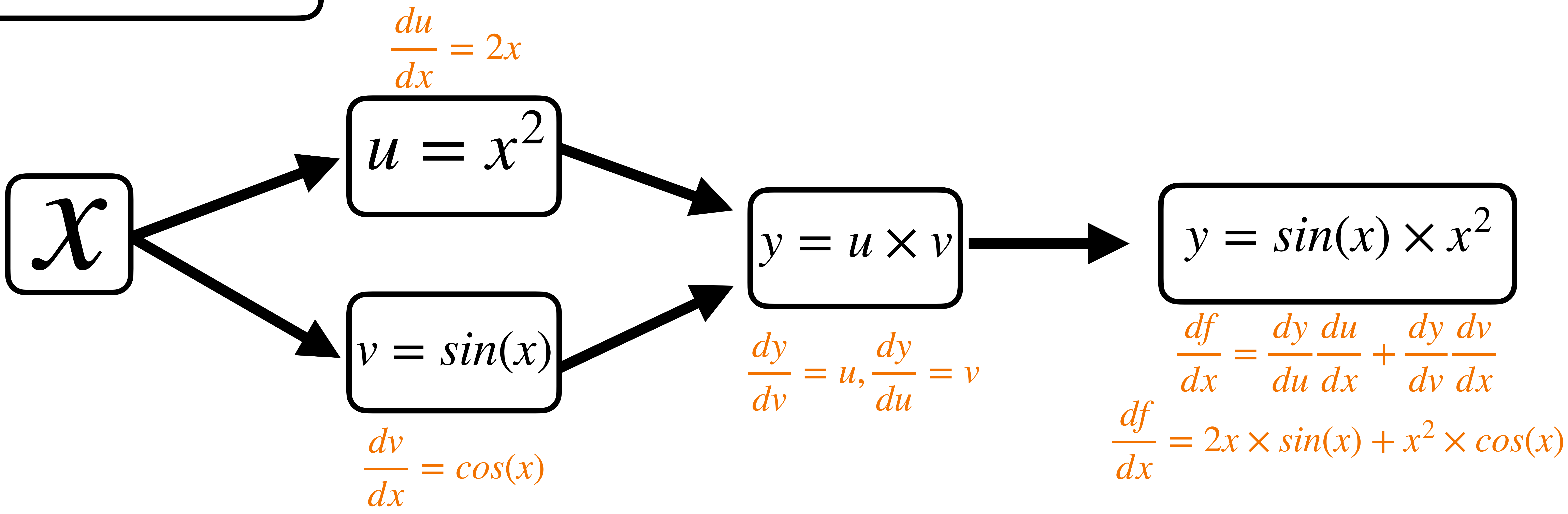
Computational Graph

Automatic Differentiation (Autodiff, AD, etc.)

i.e., the chain-rule!

$$f(x) \to y$$

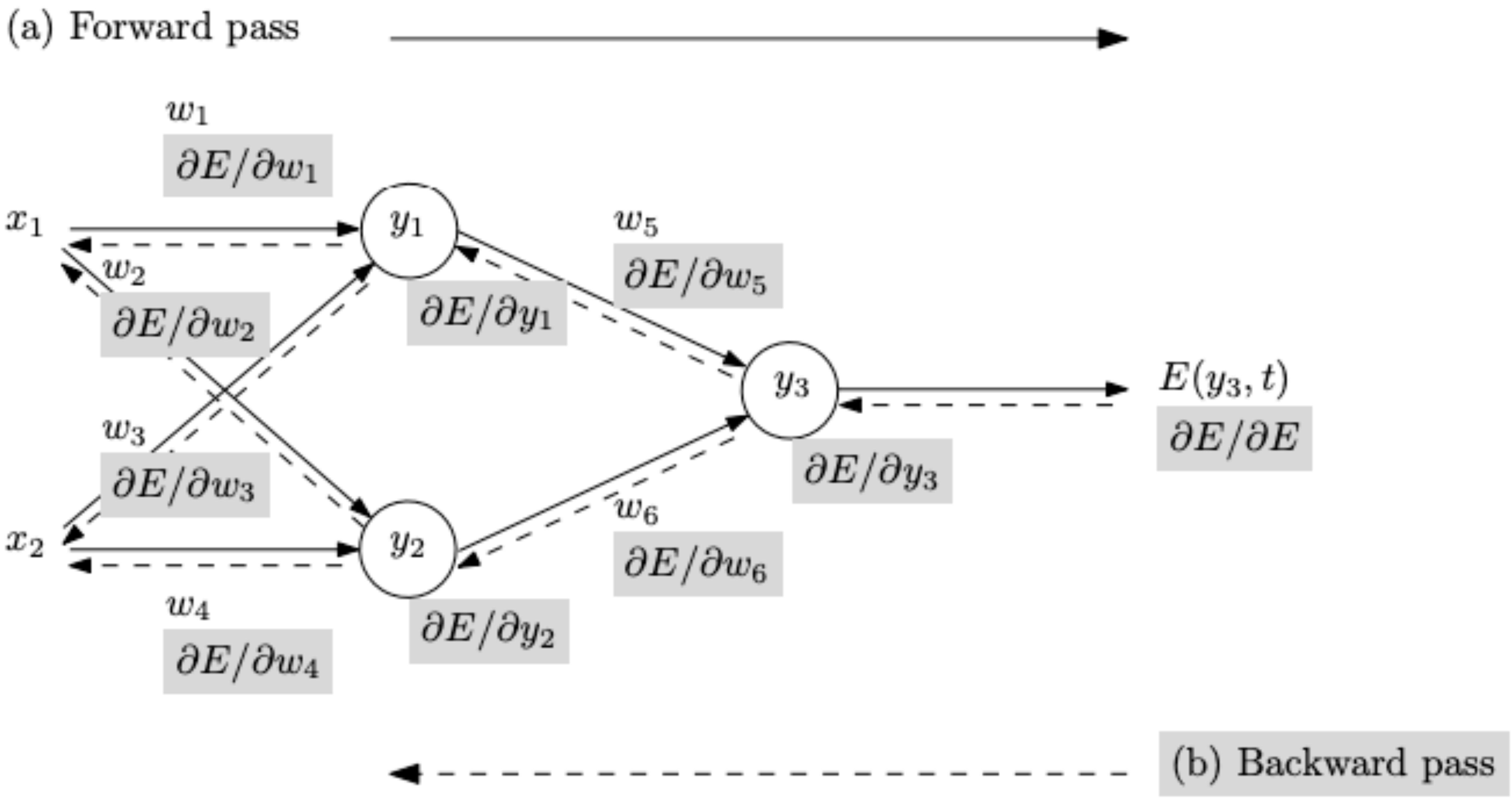
$$\boxed{y = sin(x) \times x^2}$$



Now scale this to millions of operations!

Two Flavors of AD:

$$f(x_1, x_2) = E$$



<u>Rule of Thumb:</u>

Forwards: $f : R^n \rightarrow R^m, n \ll m$

Backwards: $f : R^n \rightarrow R^m, n \gg m$

Credit: <u>https://gowrishankar.info</u>

Where can I use AD in astronomy?

- Physics & Motion Simulations (e.g., N-body)
- Solving complex non-linear PDEs (e.g., Fluid Dynamics)
- Verification of code correctness through derivatives
- Information Theory (e.g., Fisher information of system)
- Probabilistic & Bayesian Inference (see you next week)
- Complex, high-order, non-linear optimization & regression (AI / ML)

Example: Optimization of large models

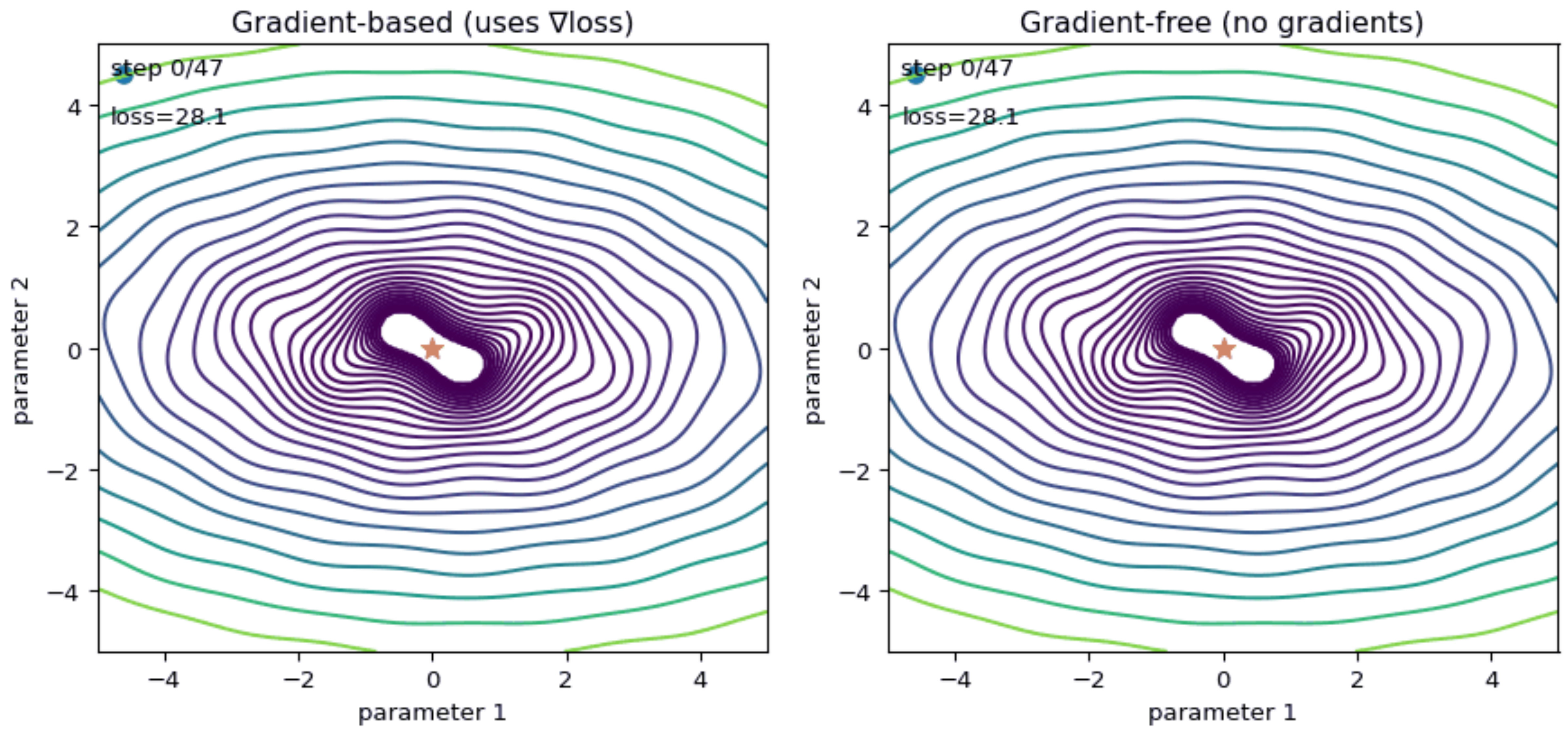
Fit a complex model with lots of parameters to some data!

Loss Function: what you are trying to optimize in regression (e.g., mean-squared error)

<u>Regression Approaches:</u>

◆Gradient-free Methods (e.g., Nelder-Mead, Simulated Annealing, LM)

◆Gradient-based Methods (e.g., SGD, L-BFGS, Adam)



Example: Optimization of large models

Gradient-free Methods

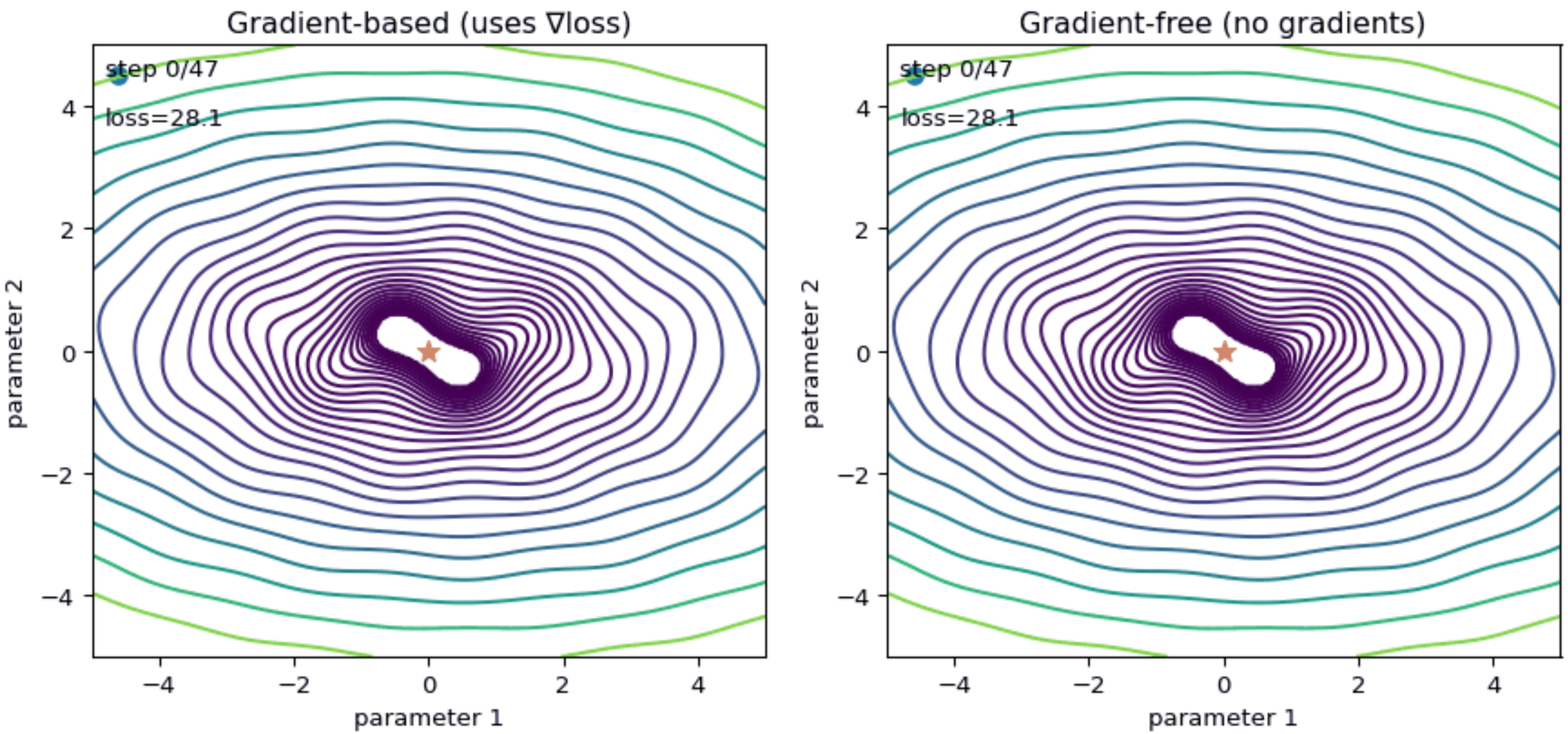
- Explores Loss Function without a priori knowledge to find optimal solution
- Optimization Scales ~ $O(N^2)$

Gradient-based Methods

- Uses partial derivatives of loss to decide how to navigate to the optimal solution
- Optimization Scales ~ $O(N)$

State-of-the-art LLM: 100's billions - trillions of free parameters

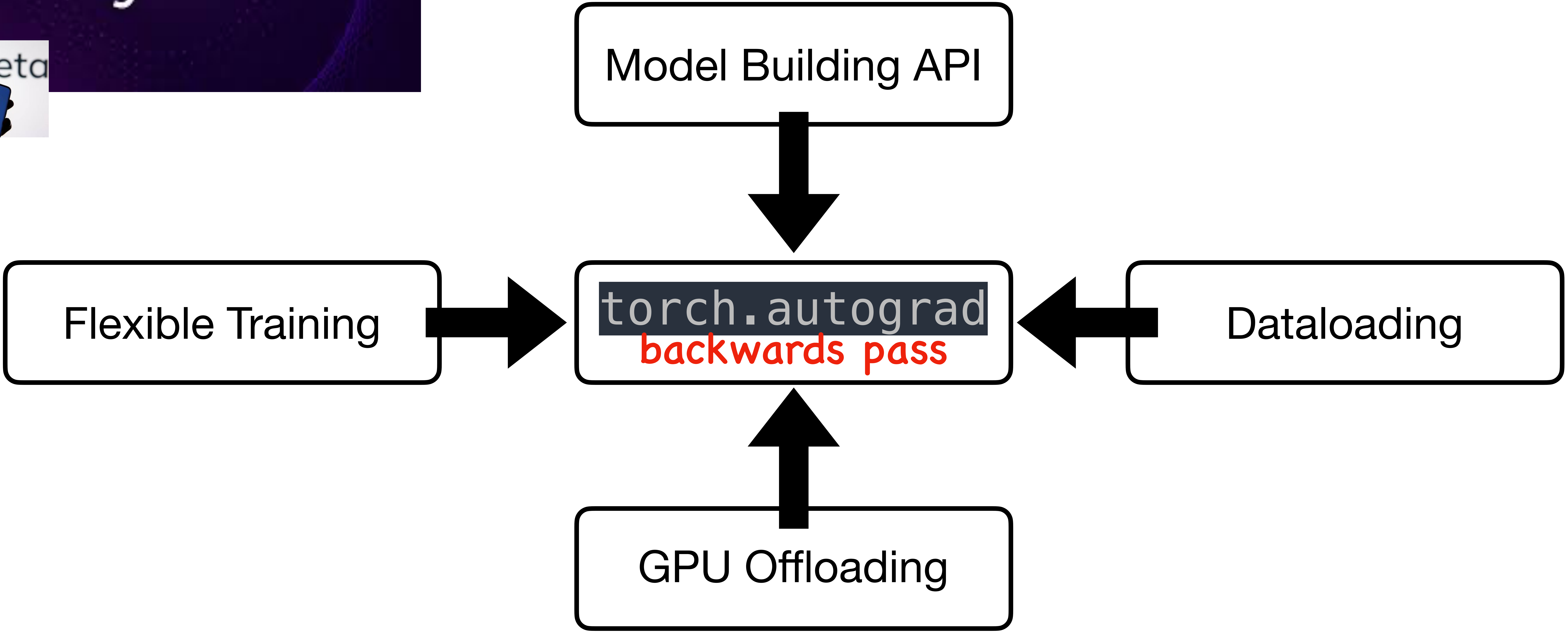
Any questions about AD?



`torch.autograd`

backwards pass

PyTorch Ecosystem



NASA-STIG — AD / PyTorch / JAX

Accelerated Computation with Auto-differentiation

Phill Cargile
Research Scientist
Center for Astrophysics | Harvard & Smithsonian
Dec. 22nd 2025

Any questions from last time?

JAX

Google

PyTorch: “Efficiently build and train AI/ML models”

JAX: “Write code (math) in a fast, differentiable, and composable way”

<u>What is so great about JAX:</u>

- ◆Automatic Differentiation (grad)
- ◆Just-in-Time Compilation (jit)
- ◆Automatic Vectorization/Parallelization (vmap/pmap)
- ◆Native GPU/TPU Offloading

JAX: “Instead of optimizing function, just improve them using transformations”

JAX

Behind the scene:

JAX -> XLA

(NumPy -> BLAS/LAPACK)

- Compilation of Computation Graphs: Compiles computation graphs into efficient machine code, needed for AD.
- Optimization Techniques: Applies operation fusion, memory optimization, and other techniques.
- Hardware Support: Optimizes models for various hardware, including CPUs, GPUs, and NPUs.
- Native access to XLA operations through .lax
- But simple NumPy/SciPy wrappers around XLA

JAX is the building block for an growing ecosystem

- Neural networks (e.g. Flax, Equinox, and Keras + JAX).
- Optimizers and solvers (e.g. Optax, Optimistix, Lineax, and Diffrax).
- Dataloading (e.g. Grain, TensorFlow Datasets, and Hugging Face Datasets).
- Probabilistic programming (e.g. Blackjax, NumPyro, and PyMC).
- Probabilistic modeling (e.g. TensorFlow Probability and Distrax).
- Physics and simulation (e.g. JAX MD and Brax).
- LLMs (e.g. MaxText, AXLearn, Levanter, and EasyLM).
- Miscellaneous coding tools (e.g. Orbax and Chex).

Final Thoughts…

PyTorch:

Pros:

- Very approachable learning curve
- Dynamic, Pythonic control flow
- Strong community and tooling for deep learning
- Debugging is straightforward

Cons:

- Autodiff is more implicit
- Less natural for non-neural-network math
- Limited inference and advanced sampling

JAX:

Pros:

- Autodiff is explicit and composable
- JIT + vectorization give high performance with limited code
- Excellent foundation for optimization and Bayesian inference
- Scales naturally to CPUs, GPUs, and TPUs

Cons:

- Steeper learning curve
- Different mindset: transformation of functions
- Debugging JIT-compiled code takes practice
- Less “plug-and-play” for beginners

Questions?

Contact me: pcargile@cfa.harvard.edu

https://astroai.cfa.harvard.edu/