Automatic Differentiation in ML

NIPS'18: Proceedings of the 32nd International Conference on Neural Information Processing Systems

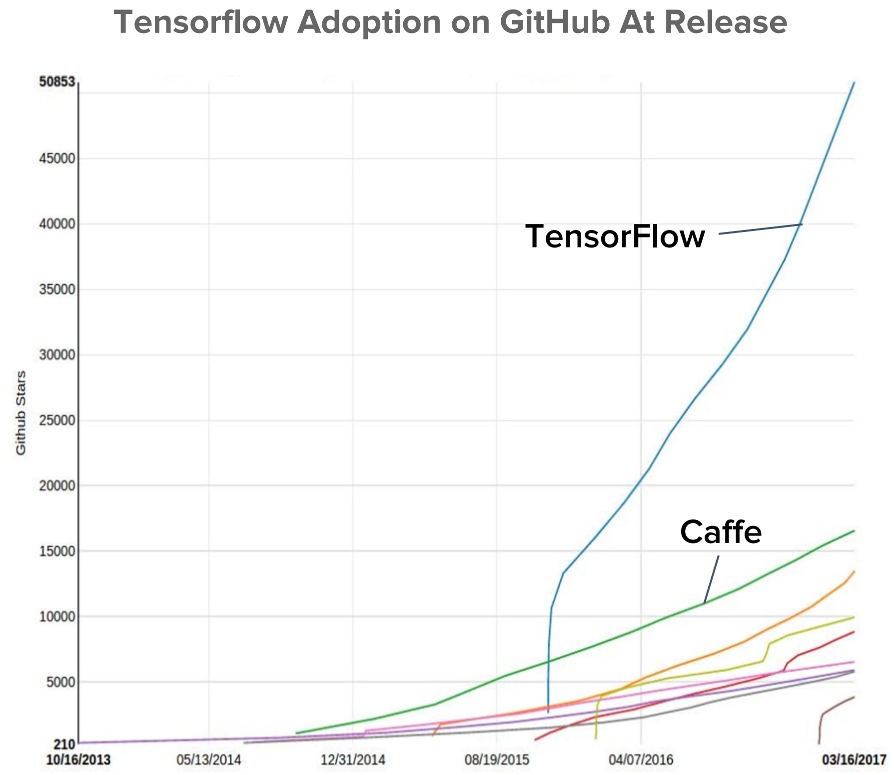
Navya Annapareddy

DS 7406

September 7th, 2022

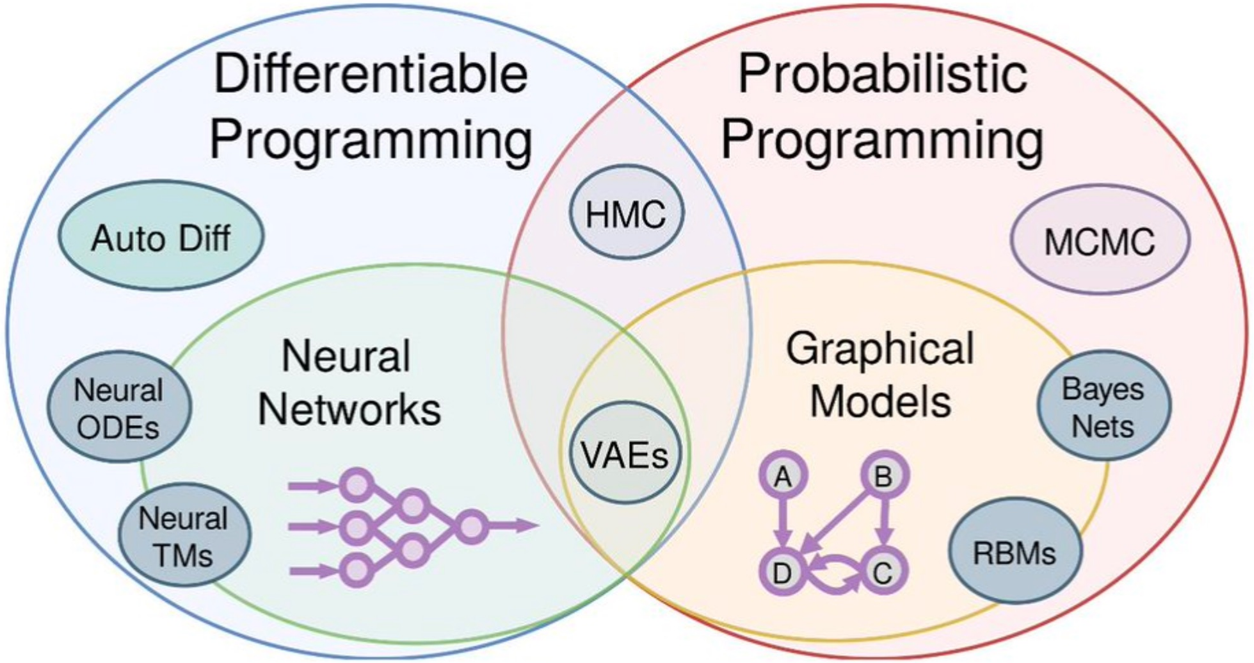
Background

- **Google Brain** established in 2011 and releases **TensorFlow** in 2015
  - Reimplemented several Theano features with more flexibility in computing (ie: GPUs)
  - Differentiable programming
- In 2018, **Google Brain** and **Quebec AI Institute**, as creators of Tensorflow and Theano, formed task force to accelerate research into automatic differentiation



Differential Programming

- Probabilistic models use MCMC/differential inference to approximate **probability distributions**
- Differentiable programming uses auto differentiation and gradient methods to approximate **loss functions**



The Role of Differentiation in ML Systems

- Backpropagation is a form of auto differentiation!
  - Networks can be represented as matrix products
  - Matrix operations can be calculated manually for gradient descent
  - Instead, autodiff programmatically calculates and optimizes gradients (partial derivatives) for weights in a given network
- Gradient descent then updates existing parameters in response to the gradients
- Most frameworks have built in AD solutions that are compiler optimized at the cost of user flexibility

Problems Solved

1. A tradeoff between **auto differentiation** and **generalizability** as well as **flexibility** exists
2. A tradeoff between **auto differentiation** and **high performance** computation exists
3. Disconnect between development of auto differentiation and other parts of ML frameworks
   a. Language design
   b. Optimizations
      i. Functional programming optimizations already adopted in ML frameworks

Problem Formulation

Methods

Review prior work and alternative approaches to AD as well as **performance, usability, and expressiveness**

Main Objective

Provide proof of concept of **high level AD framework** that resolves tradeoffs of flexibility and performance (key metrics)

Reverse Automatic Differentiation

- Chain rule for partial derivatives can evaluate
  - Forward (left to right)
    - Straightforward
    - Constant memory
    - Runtime proportional to inputs
  - Reverse (right to left)
    - Complex but utilized because inputs > outputs
      - Primal program obtains output
      - Adjoint program is run to compute gradient going backwards from output
      - Each statement in the adjoint needs access to intermediate variables of primal so they cannot be destroyed
    - Memory grows with intermediate variables
    - Runtime proportional to outputs
    - Backpropagation is specifically the application of reverse AD in ML

Approaches to Automatic Differentiation

Category	Approach	Description	Pros	Cons
Tracing	Operator Overloading	Primitive + inputs logged on “tape” to retain intermediates	• Straightforward • Simplifies AD Logic	• At runtime so not efficient
Source Transformation	General	Explicitly constructs adjoint using internal rules/control once	• Does not occur runtime so more efficient • Can be optimized before	• Requires un/parsers, interpreters
	Tape Based	Global data structure to store intermediates	• Forward pass writes intermediates and is read during backward pass	• Variable at runtime • Codependence on intermediates
	Closure Based	Eliminate custom compiler passes	• High efficiency	• Currently proof of concept
Existing Dataflows	Graph Representations	Use computation graphs as intermediate representation	• Non recursive • No tapes/closure needed because forward pass is accessible globally	• Non recursive

Proposal

- Proposed new approach of automatic differentiation called **Myia** with ideal characteristics

Graph based

Introduces **recursive graphs** to minimize tradeoffs

Closure transformation

Introduces functional programming to connect **closures**, joint optimizable

Evaluation

- Preprint on Arvix in 2018
- Accepted at NeurIPS 2019
- Myia implemented in 2020
  - Parser
    - Internal rules
  - Intermediate representations
  - Primitives
    - ie: map, reduce
  - Optimization
    - ie: closure chaining

UViM: A Unified Modeling Approach for Vision with Learned Guiding Codes

Alexander Kolesnikov[*†] **André Susano Pinto**[*†]
Lucas Beyer[*] **Xiaohua Zhai**[*] **Jeremiah Harmsen**[*] **Neil Houlsby**[*]
Google Research, Brain Team Zürich
{akolesnikov,andresp,lbeyer,xzhai,jeremiah,neilhoulsby}@google.com

PyTorch Autograd

Long Term Impact

- Support for optimizations ongoing
- “Commoditized” advanced AD in ML community
- Current state of AD in primary ML frameworks

Popularity

Category	Approach	Myia	Theano	Tensorflow	PyTorch
Tracing	Operator Overloading				x
Source Transformation	Tape			x	
	Closure	x			
Dataflows	Graph Representations	x	x	x	x

Questions?