Deep Learning

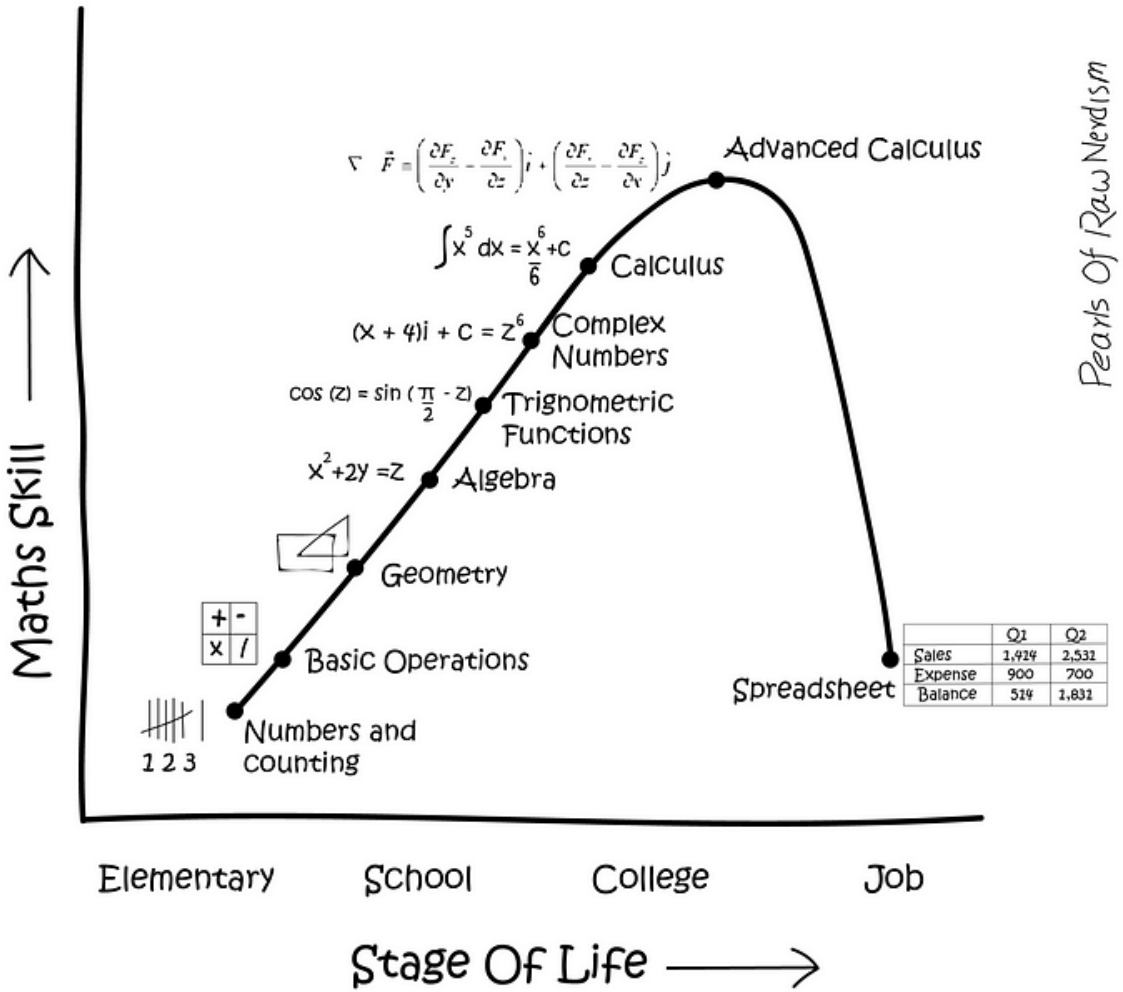
A course about theory & practice

UNIVERSITÀ DI PAVIA

Differentiating Algorithms?

Marco Piastra

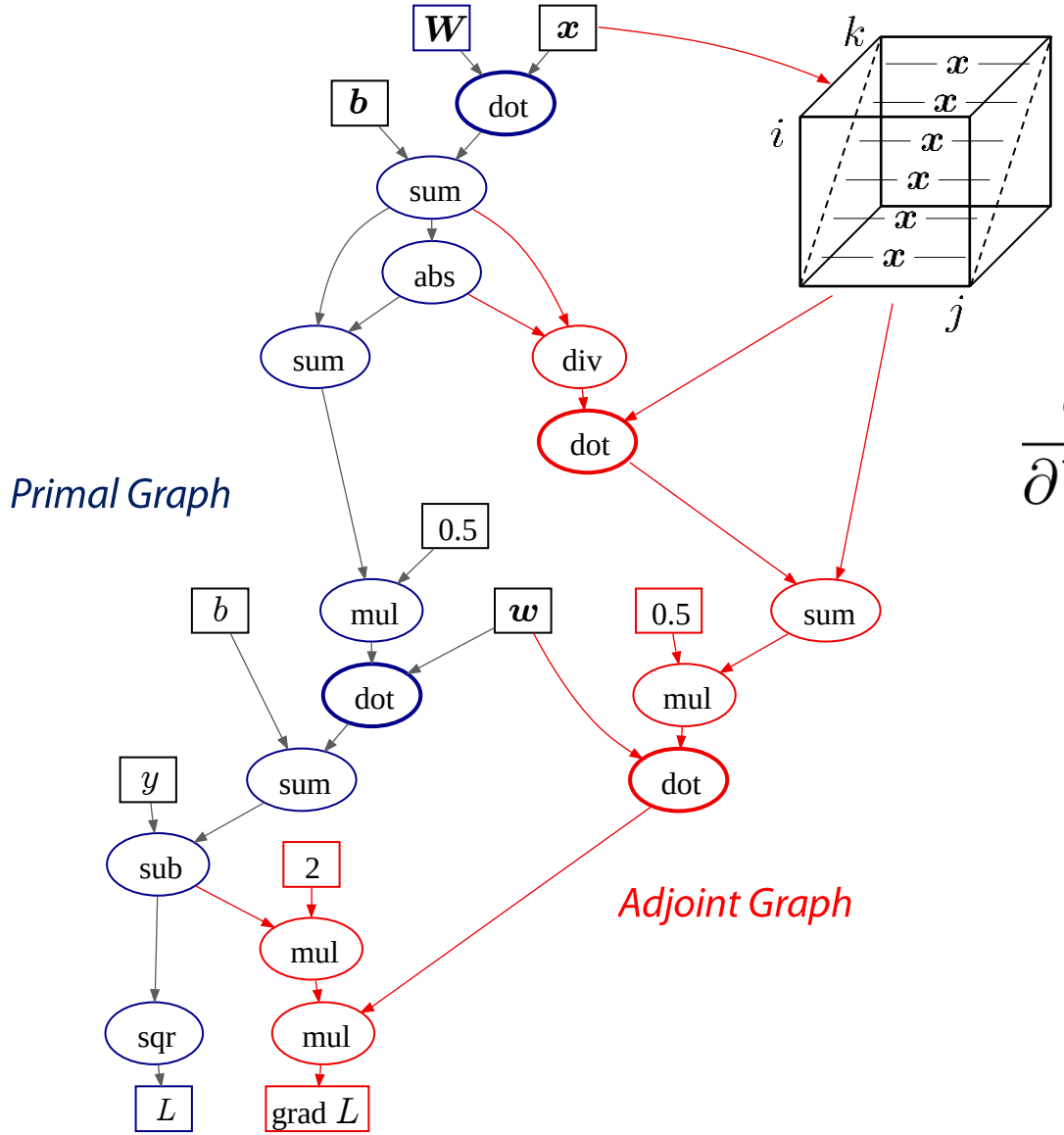
Aside the Aside



[Image from: https://medium.com/passivelogic/intro-to-differentiable-swift-part-0-why-automatic-differentiation-is-awesome-a522128ca9e3]

Graph-Based Automatic Differentiation

Automatic Differentiation (AD): Graph-Based



$$\frac{\partial}{\partial \boldsymbol{W}}(\boldsymbol{w} \cdot \mathrm{ReLU}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b - y)^2$$

AD of Flow Control Structures

- Differentiating Any Functions

```
def pow(x, n):
    r = 1
    while n > 0:
        n -= 1
        r *= x
    return r
```

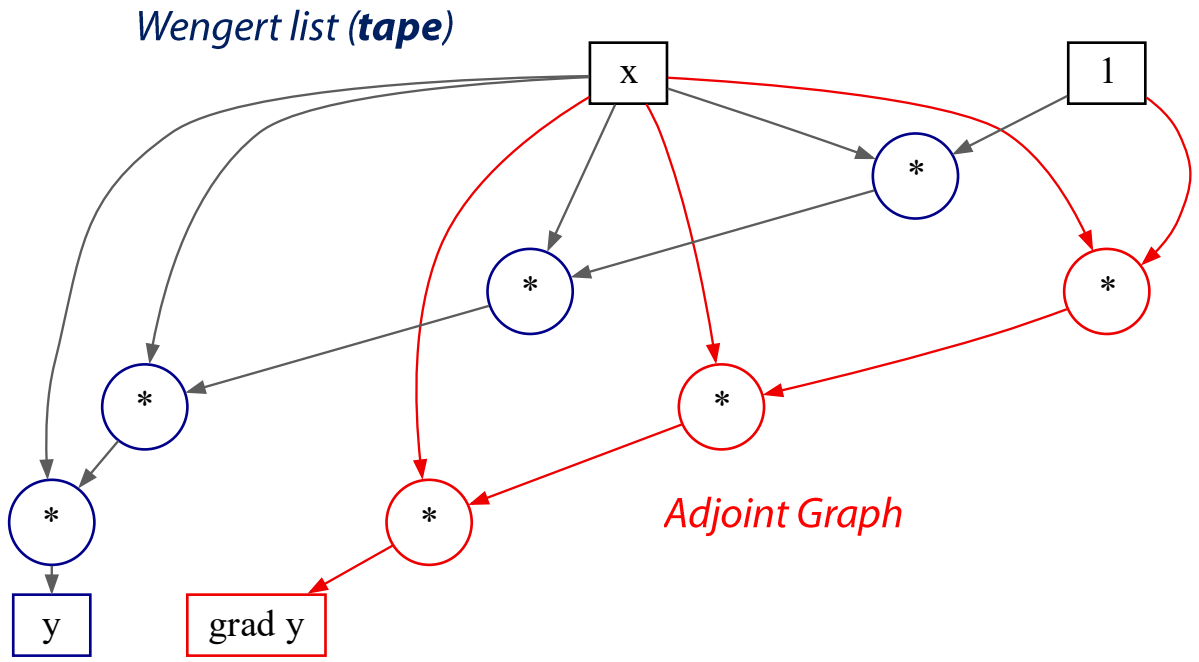
How can a `while` *structure be differentiated?*

Consider the runtime trace of a particular execution:

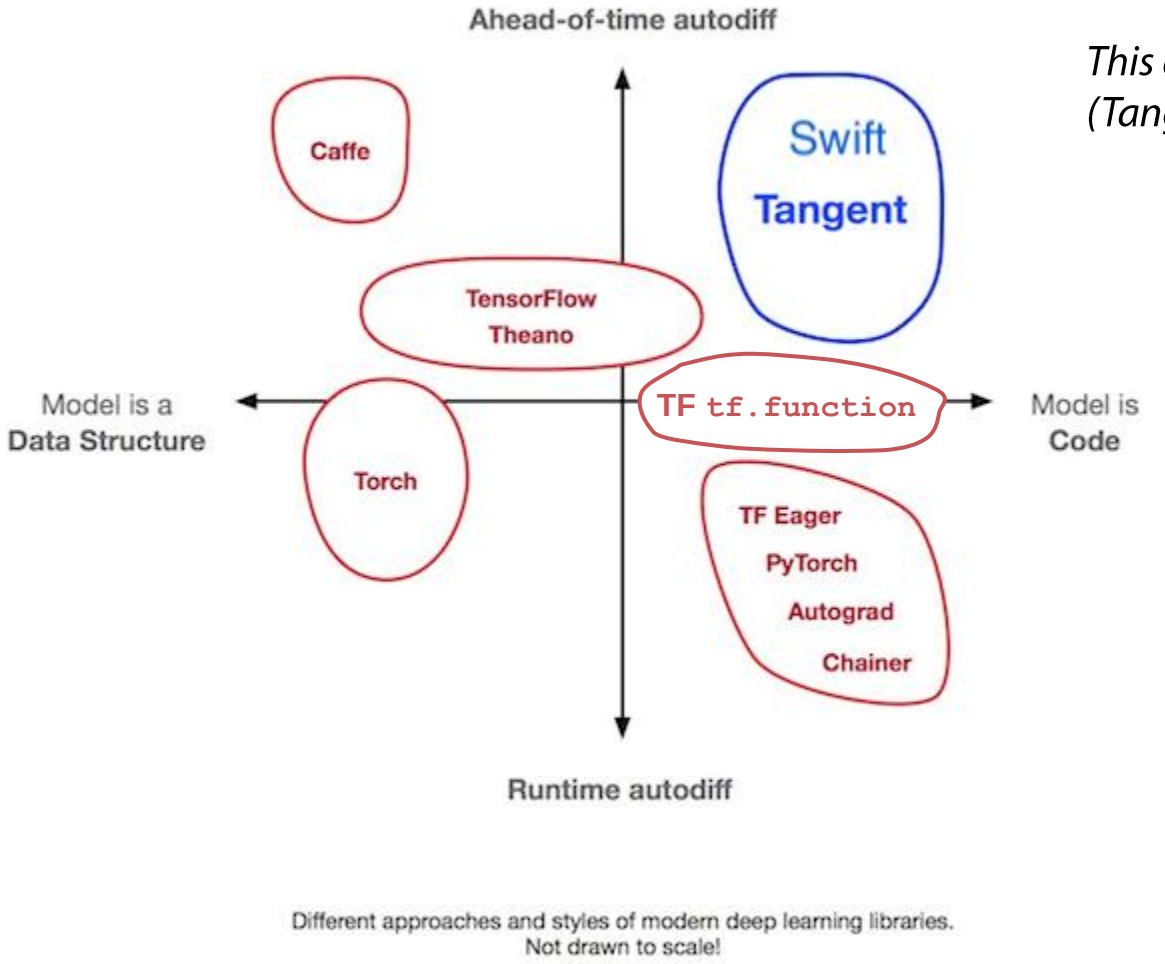
```
y = pow(x, 4)

grad(y) = ?
```

```
r = 1
r = r * x
r = r * x
r = r * x
r = r * x
y = r
```

It is also called *Wengert list*, or ***tape***



AD Frameworks



Different approaches and styles of modern deep learning libraries.
Not drawn to scale!

This diagram is a bit obsolete now
(Tangent was archived in 2021)

[Image edited from: https://github.com/tensorflow/swift/blob/main/docs/AutomaticDifferentiation.md]

AD strategies

- ***Graph-based***
  - It must be constructed explicitly, by the programmer
  - The primal graph and the adjoint graph can be both constructed once and for all
  - The combination of both graphs can be optimized as much as needed
  - Memory blocks need only be allocated at runtime and reclaimed once not used

  Programming is cumbersome and counter-intuitive (with control structures, in particular)

- ***Wengert List ('trace', 'Tape-based')***
  - It can be constructed automatically, at runtime
  - The primal graph must be collected each time, the adjoint graph needs to be computed each time and 'on the fly'
  - Optimization introduces a runtime overhead: apply with care
  - Memory in the primal graph needs be kept allocated until the gradients are computed

  Programming is only slightly different from normal; control structures can be used as usual

Different Frameworks: Engineering Trade-Offs

- **TensorFlow 1.x**

  Construction of static graphs, using a separate language (define-*and*-run)

- **PyTorch 1.x**

  "*Eager by design*"

  Overloading of Python operators, trace operation on tensors (define-*by*-run)
  Autodiff in backward mode

- **TensorFlow 2.x**

  Eager mode (no `@tf.function` *decorator)*

  Overloading of Python operators, trace operation via tape (define-*by*-run)

  Graph-based, using `@tf.function`

  Decorated function are translated once, on their first execution, into a graph-builder (define-*and*-run)
  Tracing via tape becomes easier (define-*by*-run)

- **PyTorch 2.x**

  Ahead-of-Time (AOT) autodiff on intermediate representation (IR). Mixed forward-backward model.

Multiple Frameworks, Right Now

TensorFlow

PyTorch

BUSINESS INSIDER

ENTERPRISE

Google is quietly replacing the backbone of its AI product strategy after its last big push for dominance got overshadowed by Meta

Matthew Lynley Jun 17, 2022, 9:40 PM CEST

Share | Save

Google CEO Sundar Pichai speaking during a Google event in California in 2016. Justin Sullivan/Getty Images

- **Google was a trailblazer in machine learning, releasing one of the first general-use frameworks.**
- **TensorFlow has since lost the hearts and minds of developers to Meta's AI framework, PyTorch.**
- **Google is now betting on a new AI project internally to replace TensorFlow called JAX.**

[Image from: https://www.businessinsider.com/facebook-pytorch-beat-google-tensorflow-jax-meta-ai-2022-6]

A Different Approach: JAX

JAX in a nutshell

- **Automatic Differentiation (Autograd)**

  https://github.com/HIPS/autograd
  JAX grad creates the gradient of any *pure* function (even plain python or Numpy)

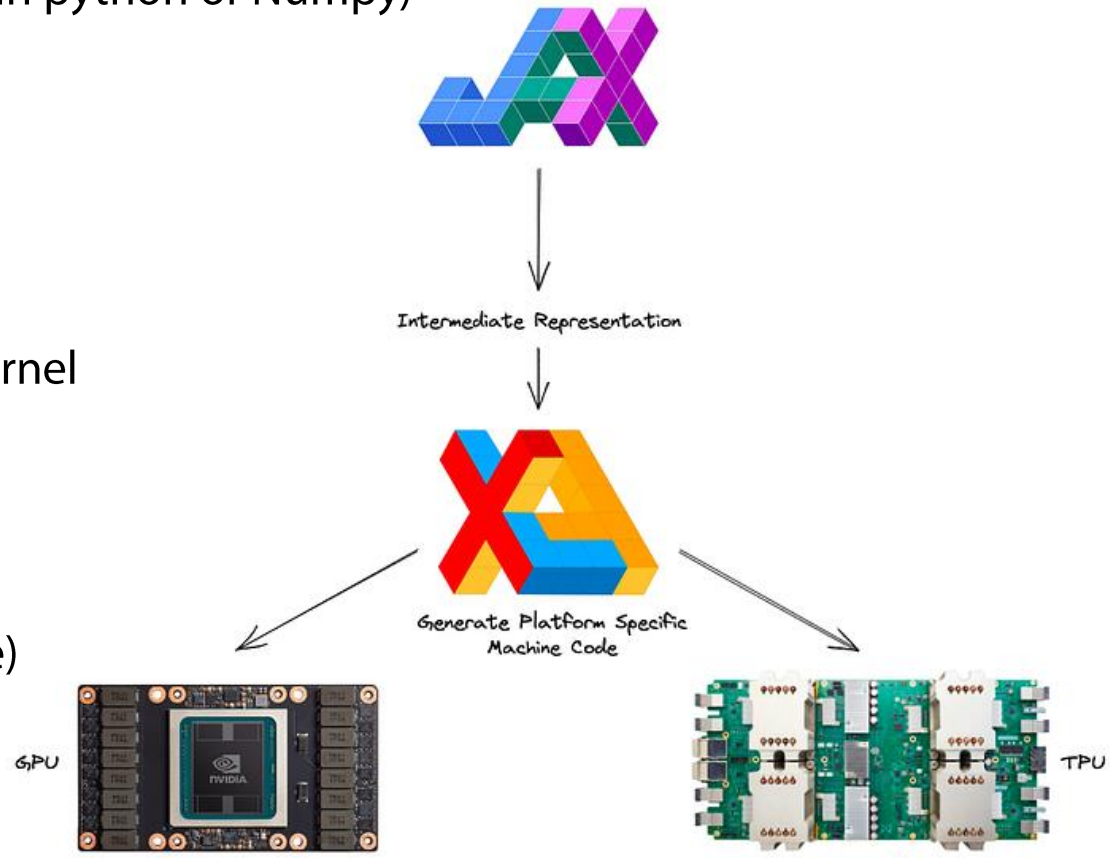
- **Automatic Vectorization**

  Transforming function that operates on a single data point
  into a function that operates on a batch of data points

- **Just-In-Time Compiler (JIT)**

  TensorFlow and PyTorch have precompiled GPU and TPU kernel
  The JAX compiler creates efficient code 'on-the-fly'

  It does not create machine code but an
  intermediate representation
  that can be further transformed
  (differentiation, vectorization, translation into machine code)



[Image from: https://www.ml6.eu/blogpost/need-for-speed-jax]