Adjoint computation and Backpropagation

Guillaume Pallez (Aupy)
Inria & University of Bordeaux

Meeting of the Royal Society, April 2019

Who's who

Automatic Differentiation

Paul Hovland (Argonne)

Navjot Kukreja (Imperial College)

Krishna Narayanan (Argonne)

Machine Learning (I)

Alexis Joly (Inria)

Machine Learning (II)

Alena Shilova (Inria)

Scheduling

Guillaume Pallez (Inria)

Olivier Beaumont (Inria)

Julien Herrmann (Inria)


THE ROYAL SOCIETY
Hierarchical Algorithms on
Hierarchical Architectures
Guillaume Pallez
Hatem Ltaief, George Turkiyyah & David Keyes
Extreme Computing Research Center (ECRC)
King Abdullah University of Science and Technology

Adjoint Computation and Backpropagation

Ice-sheet model (I)

"In climate modelling, Ice-sheet models use numerical methods to simulate the evolution, dynamics and thermodynamics of ice sheets." (wikipedia)

Model Algorithm (single timestep)

1. Evaluate driving stress $\tau_d = \rho g h \nabla s$
2. Solve for velocities

   DO i = 1, max_iter

   i. Evaluate nonlinear viscosity ν_i from iterate $\boldsymbol{u}_i$

   ii. Construct stress matrix $A\{\nu\}$

   iii. Solve linear system $A\,\boldsymbol{u}_{i+1} = \tau_d$

   iv. (Exit if converged)

   ENDDO
3. Evolve thickness (continuity eqn)

Automatic differentiation (AD) tools generate code for adjoint of operations

Credit: Daniel Goldberg

ICE-SHEET MODEL (I)

"In climate modelling, Ice-sheet models use numerical methods to simulate the evolution, dynamics and thermodynamics of ice sheets." (wikipedia)

Model Algorithm (single timestep)

1. Evaluate driving stress $\tau_d = \rho g h \nabla s$
2. Solve for velocities

   DO i = 1, max_iter

   i. Evaluate nonlinear viscosity ν_i from iterate $\boldsymbol{u}_i$

   ii. Construct stress matrix $A\{\nu\}$

   iii. Solve linear system $A\, \boldsymbol{u}_{i+1} = \tau_d$

   iv. (Exit if converged)

   ENDDO
3. Evolve thickness (continuity eqn)

Automatic differentiation (AD) tools generate code for adjoint of operations

Simpler Version:

```
proc Model Algorithm(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
    end
    /*  F(u_0) = f_n ∘ f_{n-1} ∘ ... ∘ f_0(u_0)   */
    Compute ∇F(u_0)y ;
end
```

Ice-sheet model (II)

A quick reminder about the gradient:

$$F(u_0) = f_n \circ f_{n-1} \circ \ldots \circ f_1 \circ f_0(u_0)$$

$$\begin{aligned}\boldsymbol{\nabla} F(u_0)\boldsymbol{y} &= Jf_0(u_0)^T \cdot \boldsymbol{\nabla}(f_n \circ f_1)(u_1) \cdot \boldsymbol{y} \\ &= \boxed{Jf_0(u_0)^T \cdot Jf_1(u_1)^T \cdot \ldots \cdot Jf_{n-1}(u_{n-1})^T \cdot Jf_n(u_n)^T \cdot \boldsymbol{y}}\end{aligned}$$

$$\begin{aligned} Jf^T &= \text{Transpose Jacobian matrix of } f; \\ u_{i+1} &= f_i(u_i) = f_i\left(f_{i-1} \circ \ldots \circ f_0(u_0)\right). \end{aligned}$$

A better solution?

$$\nabla F(u_0)\boldsymbol{y} = Jf_0(u_0)^T \cdot Jf_1(u_1)^T \cdot \ldots \cdot Jf_{n-1}(u_{n-1})^T \cdot Jf_n(u_n)^T \cdot \boldsymbol{y}$$

```
proc Algo A(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
    end
    Compute ∇F(u_0)y;
end
```

A better solution?

$$\nabla F(u_0)\boldsymbol{y} = Jf_0(u_0)^T \cdot Jf_1(u_1)^T \cdot \ldots \cdot Jf_{n-1}(u_{n-1})^T \cdot Jf_n(u_n)^T \cdot \boldsymbol{y}$$

```
proc Algo A(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
    end
    Compute ∇F(u_0)y;
end
```

→ What is the problem with Algo A?

A better solution?

$$\nabla F(u_0)\boldsymbol{y} = Jf_0(u_0)^T \cdot Jf_1(u_1)^T \cdot \ldots \cdot Jf_{n-1}(u_{n-1})^T \cdot Jf_n(u_n)^T \cdot \boldsymbol{y}$$

```
proc Algo A(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
    end
    Compute ∇F(u_0)y;
end
```

→ What is the problem with Algo A?

```
proc Algo B(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
        v_{i+1} = v_i · Jf_{i+1}(u_{i+1})^T;
    end
end
```

A better solution?

$$\nabla F(u_0)\boldsymbol{y} = Jf_0(u_0)^T \cdot Jf_1(u_1)^T \cdot \ldots \cdot Jf_{n-1}(u_{n-1})^T \cdot Jf_n(u_n)^T \cdot \boldsymbol{y}$$

```
proc Algo A(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
    end
    Compute ∇F(u_0)y;
end
```

→ What is the problem with Algo A?

```
proc Algo B(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
        v_{i+1} = v_i · Jf_{i+1}(u_{i+1})^T;
    end
end
```

→ What is the problem with Algo B?

A better solution?

$$\nabla F(u_0)\boldsymbol{y} = Jf_0(u_0)^T \cdot Jf_1(u_1)^T \cdot \ldots \cdot Jf_{n-1}(u_{n-1})^T \cdot Jf_n(u_n)^T \cdot \boldsymbol{y}$$

```
proc Algo A(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
    end
    Compute ∇F(u_0)y;
end
```

→ What is the problem with Algo A?

```
proc Algo B(u_0, y)
begin
    Do stuff;
    for i = 0 to n do
        u_{i+1} = f_i(u_i);
        Do stuff;
        v_{i+1} = v_i · Jf_{i+1}(u_{i+1})^T;
    end
end
```

→ What is the problem with Algo B?

$$\nabla F(u_0)\boldsymbol{y} = \left(\left(\ldots \left(Jf_0{}^T \cdot Jf_1{}^T\right) \cdot \ldots \cdot Jf_{n-1}{}^T\right) \cdot Jf_n{}^T\right) \cdot \boldsymbol{y} \qquad n \text{ MatMat ops}$$

$$\nabla F(u_0)\boldsymbol{y} = Jf_0{}^T \cdot \left(Jf_1{}^T \cdot \ldots \cdot \left(Jf_{n-1}{}^T \cdot \left(Jf_n{}^T \cdot \boldsymbol{y}\right) \ldots\right)\right) \qquad n \text{ MatVec ops}$$

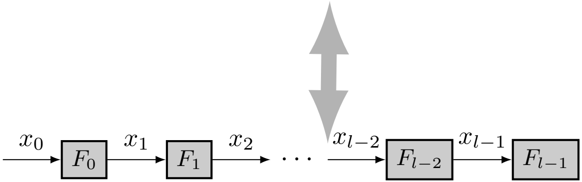
Adjoint computation

$$\begin{aligned} F_i(x_i) &= x_{i+1} \quad & i < l \\ \bar{F}_i(x_i, \bar{x}_{i+1}) &= \bar{x}_i \quad & i \leq l \end{aligned}$$

Adjoint computation

$$\boldsymbol{F_i(x_i) = x_{i+1} \quad i < l}$$

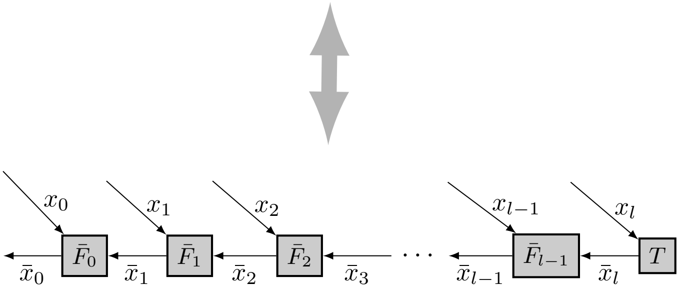
$$\bar{F}_i(x_i, \bar{x}_{i+1}) = \bar{x}_i \qquad i \leq l$$



Adjoint computation

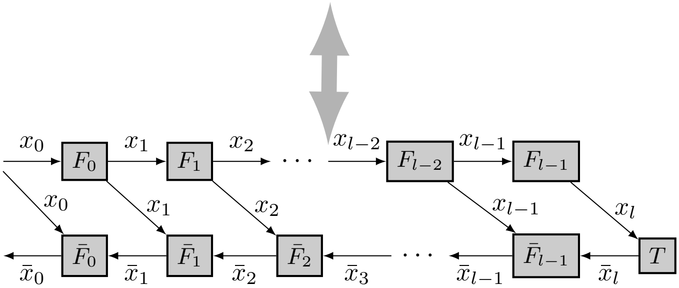
$$F_i(x_i) = x_{i+1} \quad i < l$$

$$\boldsymbol{\bar{F}_i(x_i, \bar{x}_{i+1}) = \bar{x}_i \quad i \leq l}$$

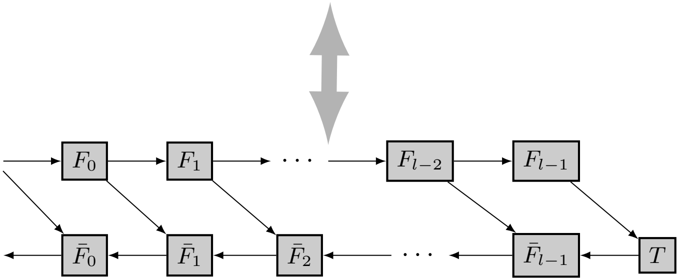


Adjoint computation

$$F_i(x_i) = x_{i+1} \quad i < l$$
$$\bar{F}_i(x_i, \bar{x}_{i+1}) = \bar{x}_i \quad i \leq l$$

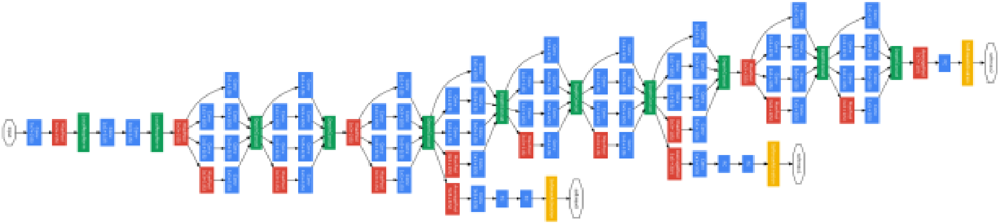


$$F_i(x_i) = x_{i+1} \quad i < l$$
$$\bar{F}_i(x_i, \bar{x}_{i+1}) = \bar{x}_i \quad i \leq l$$



Relation to IA? (I)

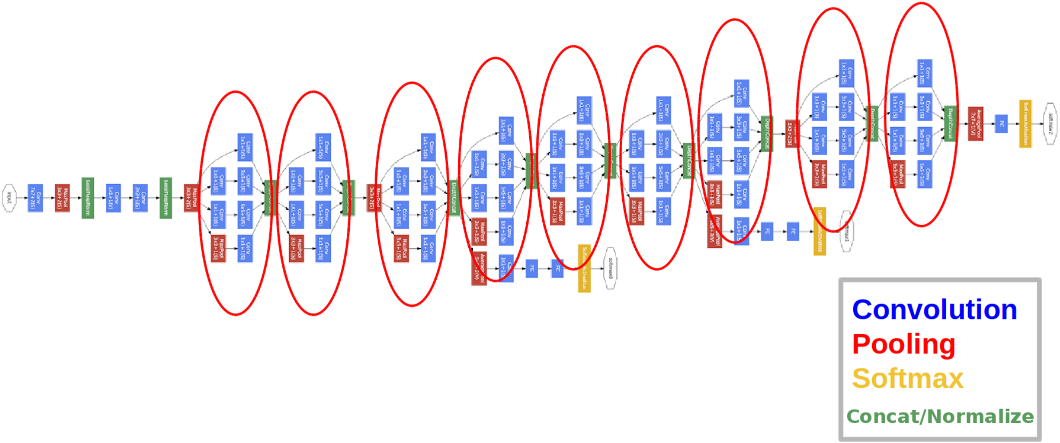
GoogleNet graph:



Source : Internet :s

Relation to IA? (I)

GoogleNet graph:

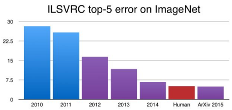


Source : Internet :s

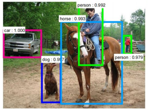
Derivatives in machine learning

"Backprop" and gradient descent are at the core of all recent advances

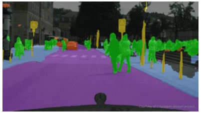
Computer vision



Top-5 error rate for ImageNet (NVIDIA devblog)

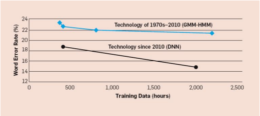


Faster R-CNN (Ren et al. 2015)



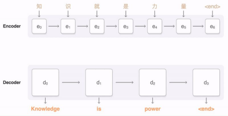
NVIDIA DRIVE PX 2 segmentation

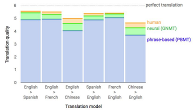
Speech recognition & synthesis



Word error rates (Huang et al., 2014)

Machine translation



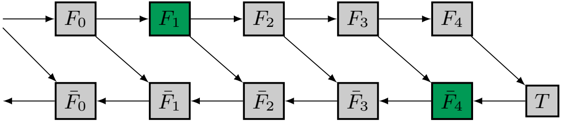


Google Neural Machine Translation System (GNMT)

4

Source: Baydin, *Beyond backprop: Automatic differentiation in machine learning*, 2017.

Model of execution

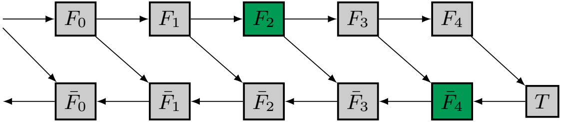


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

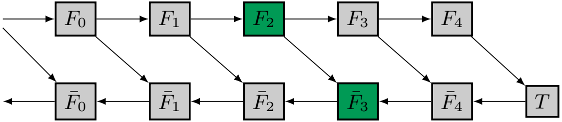


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

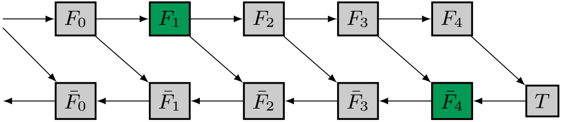


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

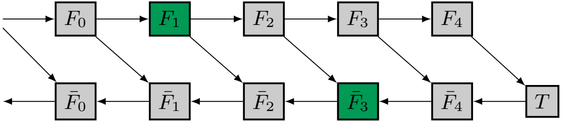


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

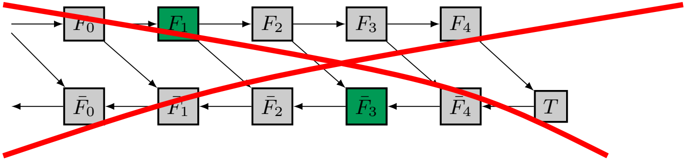


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

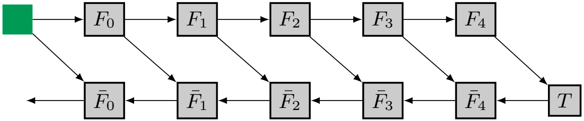


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

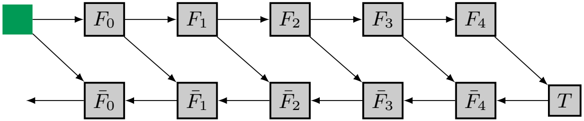


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.

Example of execution

Strategy	Time	Space

Model of execution

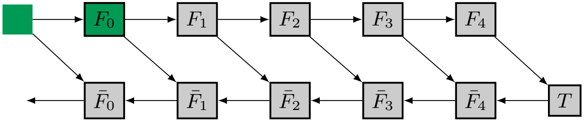


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution

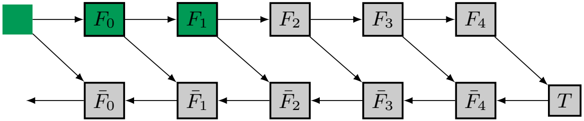


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution

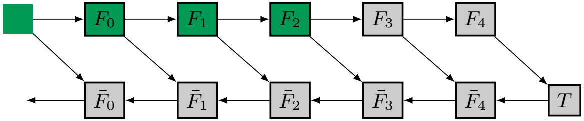


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

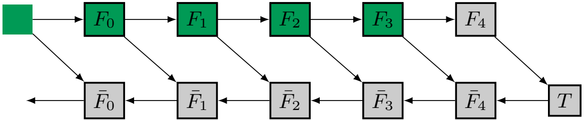
Model of execution



- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

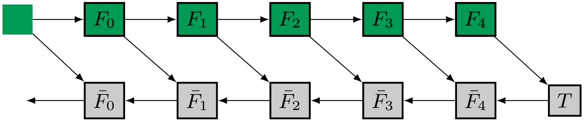


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution



Peak Mem

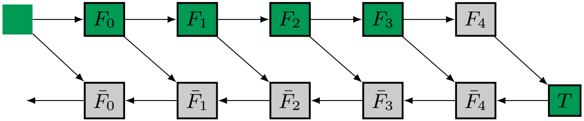
6

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution



Peak Mem

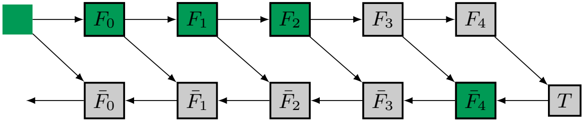
6

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution



Peak Mem

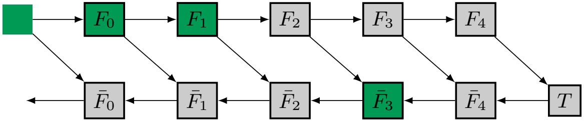
6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

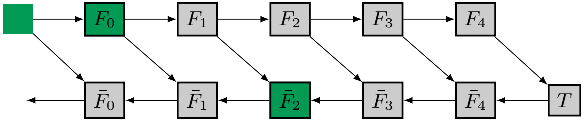
6

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution



Peak Mem

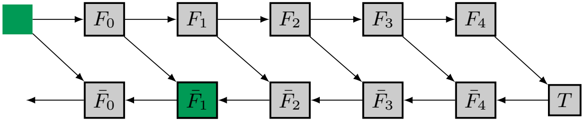
6

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution



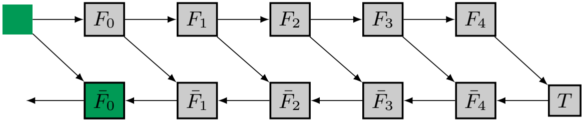
Peak Mem
6

Example of execution

Strategy	Time	Space

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

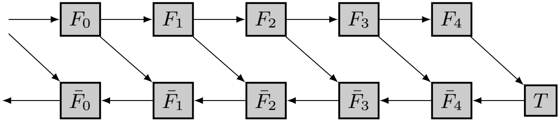
6

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space

Model of execution



Peak Mem

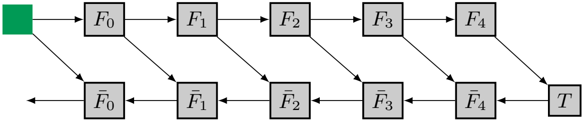
6

Example of execution

	Strategy	Time	Space
	Store all	\$	\$\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



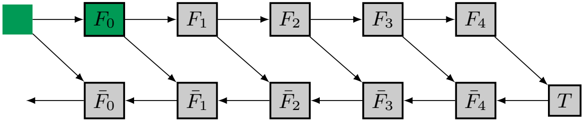
Peak Mem

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



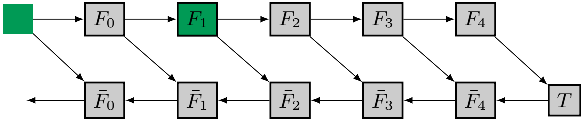
Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

Model of execution



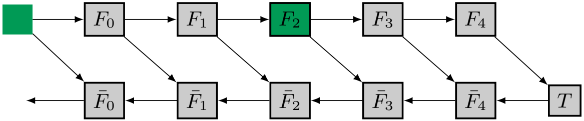
Peak Mem

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



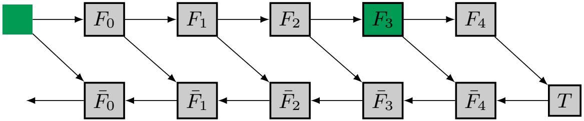
Peak Mem

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store “none”		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



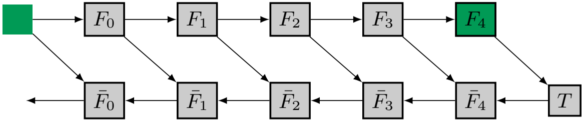
Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store “none”		

Model of execution



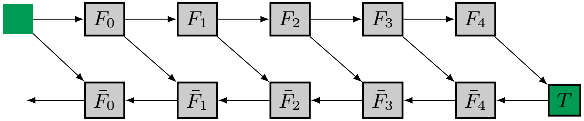
Peak Mem

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



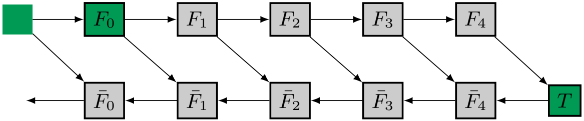
Peak Mem

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

Model of execution



Peak Mem

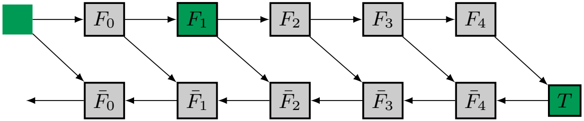
3

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	$	$$$
Store “none”		

Model of execution



Peak Mem

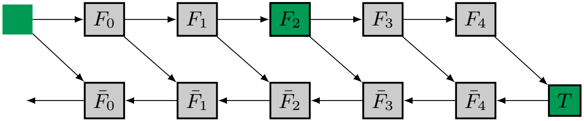
3

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

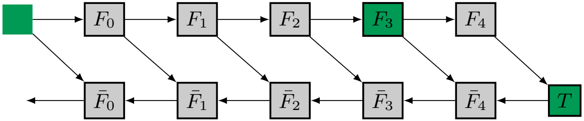
3

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"		

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

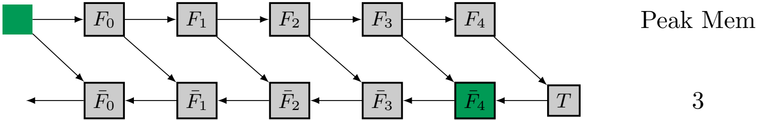
3

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”		

Model of execution

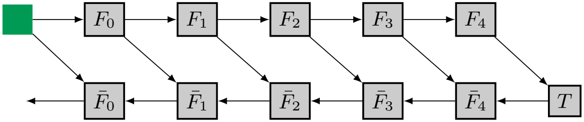


Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

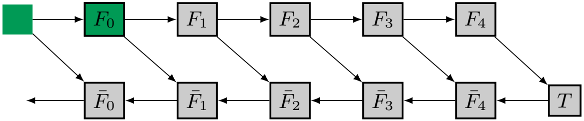
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

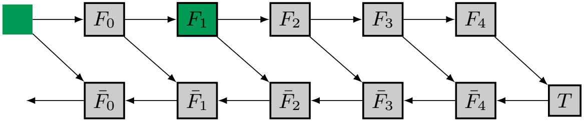
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

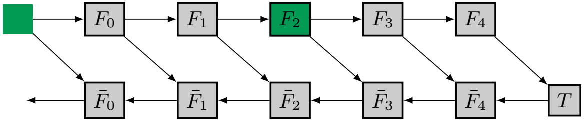
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

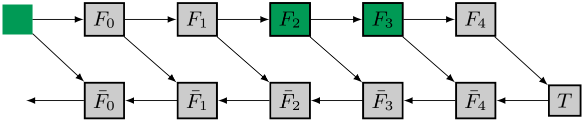
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution

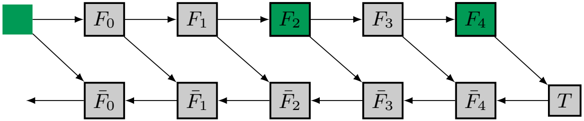


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



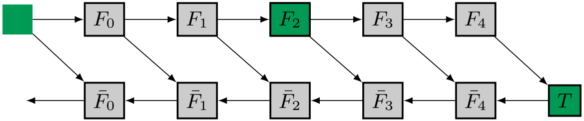
Peak Mem

4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$



Peak Mem

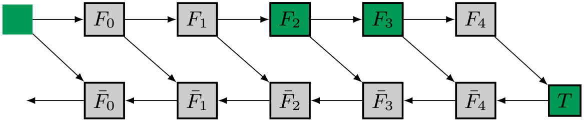
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

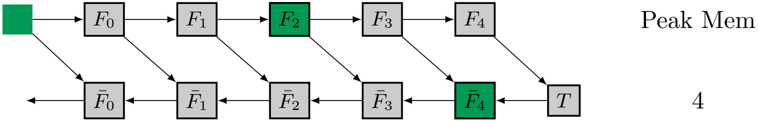
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution

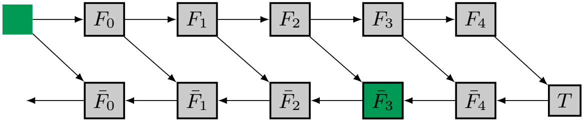


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



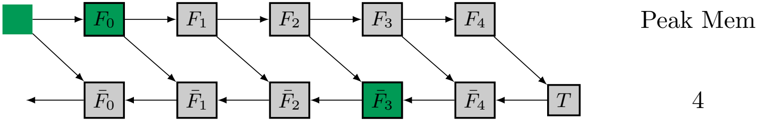
Peak Mem
4

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution

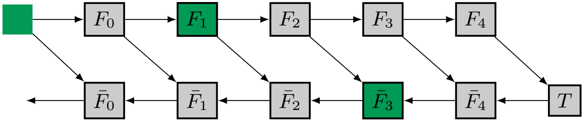


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

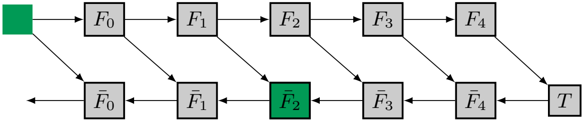
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

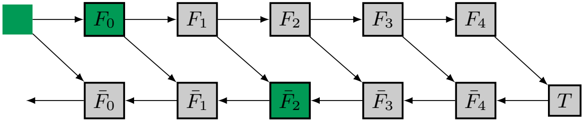
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

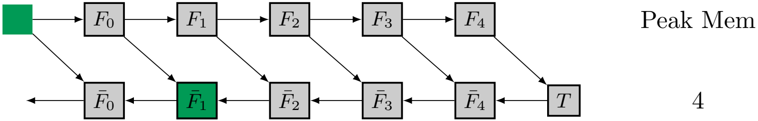
4

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution

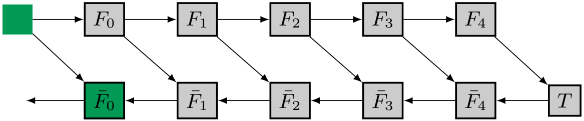


- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

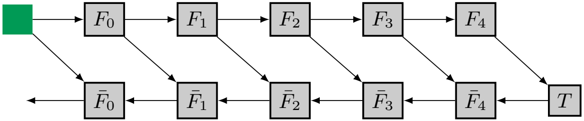
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$

Model of execution



Peak Mem

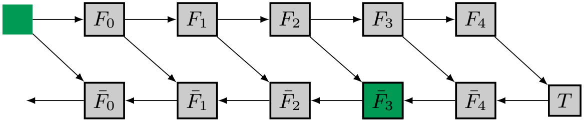
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$
Memory Reuse	\$	\$

Model of execution



Peak Mem

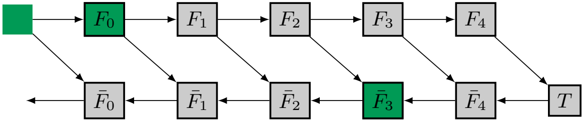
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$
Memory Reuse	\$	\$

Model of execution



Peak Mem

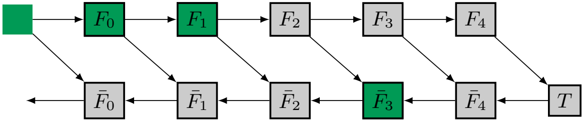
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$
Memory Reuse	\$	\$

Model of execution



Peak Mem

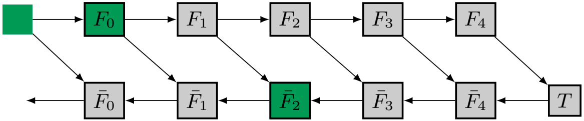
4

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store “none”	\$\$\$	\$
No reuse	\$\$	\$\$
Memory Reuse	\$	\$

Model of execution



Peak Mem

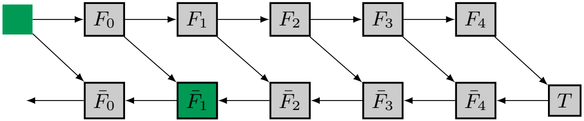
4

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$
Memory Reuse	\$	\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



Peak Mem

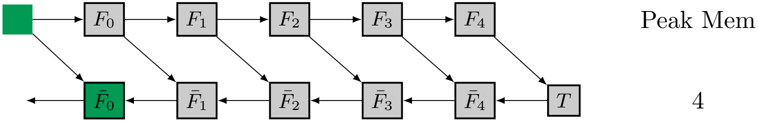
4

Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$
Memory Reuse	\$	\$

- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

Model of execution



- Memory to store output of computations (x_i or $\bar{x}_i$). Initial state: contains x_0.
  - Cost to write: $w_m = 0$,
  - Cost to read: $r_m = 0$.

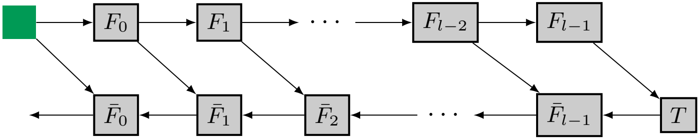
Example of execution

Strategy	Time	Space
Store all	\$	\$\$\$
Store "none"	\$\$\$	\$
No reuse	\$\$	\$\$
Memory Reuse	\$	\$

Problem formulation

We want to minimize the makespan of:

		Initial state:
AC graph:	size l	
Steps:	u_f, u_b	
Memory:	c_m, $w_m = r_m = 0$,	$\mathcal{M}_{\text{ini}} = \{x_0\}$
Storage k:	c_k, w_k, r_k,	$S_{\text{ini}} = \emptyset$



Existing work

Question: How to organize the reverse execution of intermediate steps? What do we store, what do we recompute?

- Store all: memory expensive
- Recompute all: compute expensive
- Intermediate status?

Griewand and Walther, 2000: Revolve(l, c_m), optimal algorithm with c_m memory slots.

5

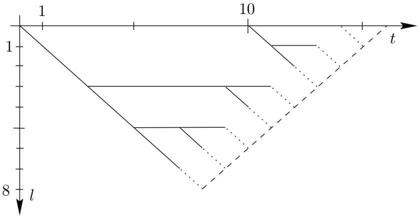


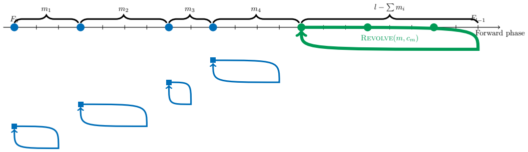
Figure 1.3: Example of Parallel Reversal Using 3 Checkpoints

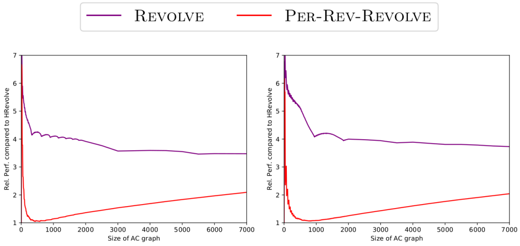
Source: Andrea Walther's PhD thesis, 1999

Storage hierarchy

A., Herrmann, Hovland, Robert, 2015: Optimal algorithm for two level of storage: cheap bounded memory and costly unbounded disks.
A., Herrmann, 2019: Library of optimal schedules for any number of storage level.

(`https://gitlab.inria.fr/adjoint-computation`)





(a) *arch_1.txt* (b) *arch_2.txt*

Fig. 5. Relative performance of the heuristics compared to the optimal solution on hierarchical platforms for large graph sizes.

What directions for AI?

Then what? Are we done? Just let AD people and ML people talk together!

Then what? Are we done? Just let AD people and ML people talk together!
Cut the middle-(scheduling)-people!

Source: A graffiti in Paris (twitter)

What directions for AI?

While the core of the algorithms remain similar, the problematics are different:

- Shallower graphs ($O(100 - 1000)$ levels).
- Cost functions (time/memory) are not necessarily uniform.
- Graphs with more structure than chains.
- Multi-Learners/Hyperparameter tuning (independent graphs executed simultaneously), shared memory?
- Etc.

What directions for AI?

While the core of the algorithms remain similar, the problematics are different:

- Shallower graphs ($O(100 - 1000)$ levels).
- Cost functions (time/memory) are not necessarily uniform.
- Graphs with more structure than chains.
- Multi-Learners/Hyperparameter tuning (independent graphs executed simultaneously), shared memory?
- Etc.

Pffiew, just saved my job

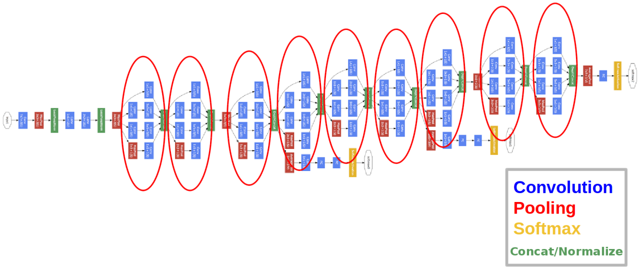
What directions for AI?

While the core of the algorithms remain similar, the problematics are different:

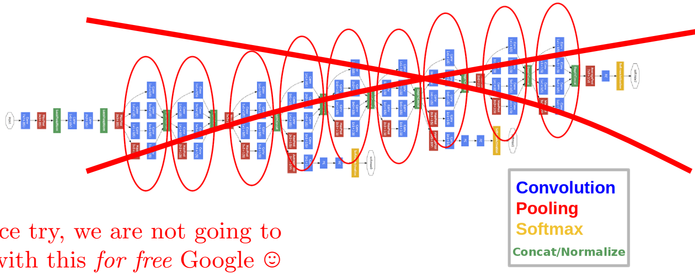
- Shallower graphs ($O(100 - 1000)$ levels).
- Cost functions (time/memory) are not necessarily uniform.
- **Graphs with more structure than chains.**
- Multi-Learners/Hyperparameter tuning (independent graphs executed simultaneously), shared memory?
- Etc.

Pffiew, just saved my job

Remember this google network:



Remember this google network:



Nice try, we are not going to help you with this *for free* Google ☺

Dir. for AI: Graph structure II

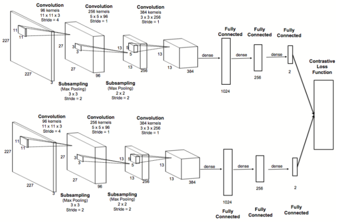


Figure 1: Siamese Neural Network Architecture

Source: Rao et al., *A Deep Siamese Neural Network (...)*, 2016

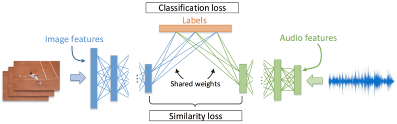
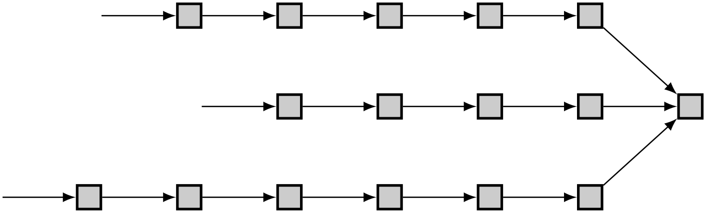


Fig. 2. Schematic of the used architecture.

Source: Surís et al., *Cross-Modal Embeddings for Video and Audio Retrieval*, 2018

Pitchfork graph[1] (aka join graphs):



Theorem (A., Beaumont, Herrmann, Shilova, 2019)

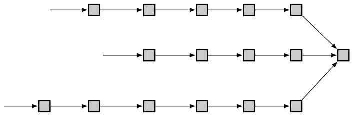
Given a bounded memory and a pitchfork with a bounded number of "teeth", we can find in polynomial time the solution that backpropagates it in minimal time.

[1] This should not be seen as an endorsement of the YJ movement, I don't think I'm allowed to give publicly my opinion.

A Grasp of the proof? (I)

Three phase algorithm:

1. Forward phase
2. Turn
3. Backward phase

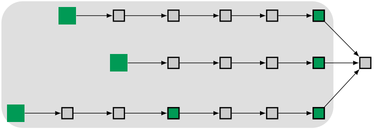


A Grasp of the proof? (I)

Three phase algorithm:

1. **Forward phase**
2. Turn
3. Backward phase

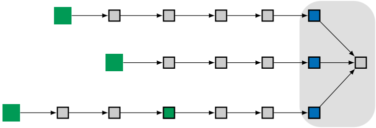
- Traverse all branches. Write some intermediate data



A Grasp of the proof? (I)

Three phase algorithm:

1. Forward phase
2. **Turn**
3. Backward phase

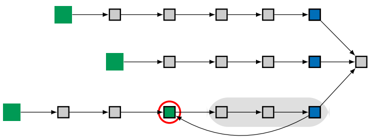


- Traverse all branches. Write some intermediate data
- *Backpropagate* the handle of the pitchfork

A Grasp of the proof? (I)

Three phase algorithm:

1. Forward phase
2. Turn
3. **Backward phase**



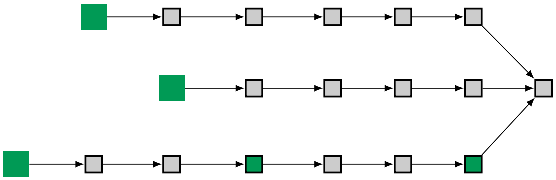
- Traverse all branches. Write some intermediate data
- *Backpropagate* the handle of the pitchfork
- Iteratively, read some checkpointed data from one of the branches, backpropagate a subset of the graph (can write additional intermediate data)

A Grasp of the proof? (II)

It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.



A Grasp of the proof? (II)

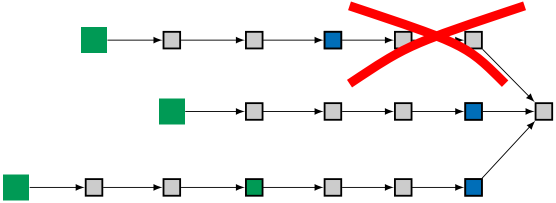
It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.

Lemma (Stability 1)

If F_i is "backpropagated", then there are no F_j for $i \leq j$.



A Grasp of the proof? (II)

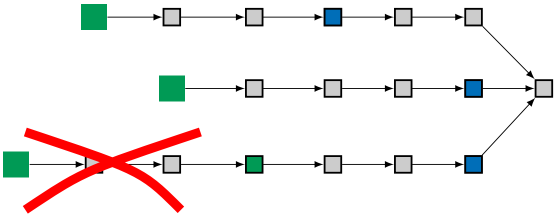
It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.

Lemma (Checkpoint persistence)

If x_i is stored, until F_i is "backpropagated", there are no F_j for $j < i$.



A Grasp of the proof? (II)

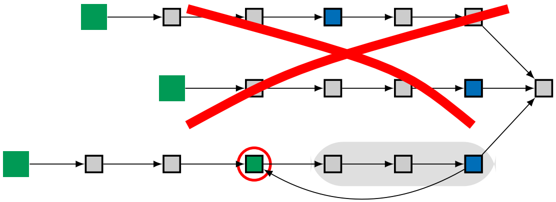
It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.

Lemma (Stability 2)

If x_i is read, then there are no F_j on other branches until it is backpropagated.



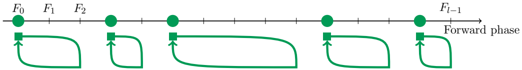
A Grasp of the proof? (II)

It relies on key properties of the **backward** phase:

- Stability of execution
- Checkpoint persistence

which give us a multi-phase approach.

In this case, for a given forward phase, we get a multi-phase backward phase:



- Where do we schedule the checkpoints in the forward phase?
- In which order do we execute the subsegment on each branch?

Is it worth it?

Is it worth it?

- From a scheduling perspective: **Yes!** (new fun problems)

Is it worth it?

- From a scheduling perspective: **Yes!** (new fun problems)

- From an adjoint perspective: **Yes!**
  With a memory of size $O(M)$:
  - *Store All* can execute a graph of size $O(M)$ in time $O(M)$;
  - *Revolve* can execute a graph of size $O(e^M)$ in time $O(Me^M)$!
  - *H-Revolve* inproves performance by a factor of magnitude.

Is it worth it?

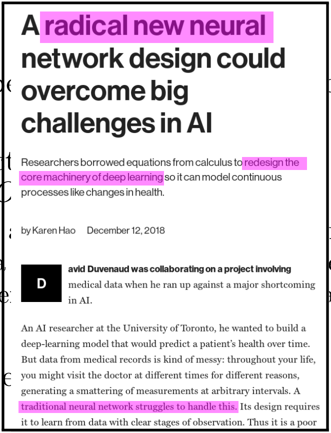
- From a scheduling perspective: **Yes!** (new fun problems)

- From an adjoint perspective: **Yes!**
  With a memory of size $O(M)$:
  - *Store All* can execute a graph of size $O(M)$ in time $O(M)$;
  - *Revolve* can execute a graph of size $O(e^M)$ in time $O(Me^M)$!
  - *H-Revolve* inproves performance by a factor of magnitude.

- Machine Learning perspective: deeper networks!

Is it worth it?

- From a scheduling perspe … oblems)
- From an adjoint perspect …
  With a memory of size C …
  - *Store All* can execute … he $O(M)$;
  - *Revolve* can execute a … e $O(Me^M)$!
  - *H-Revolve* inproves pe … agnitude.
- Machine Learning perspe …



A radical new neural network design could overcome big challenges in AI

Researchers borrowed equations from calculus to redesign the core machinery of deep learning so it can model continuous processes like changes in health.

by Karen Hao December 12, 2018

David Duvenaud was collaborating on a project involving medical data when he ran up against a major shortcoming in AI.

An AI researcher at the University of Toronto, he wanted to build a deep-learning model that would predict a patient's health over time. But data from medical records is kind of messy: throughout your life, you might visit the doctor at different times for different reasons, generating a smattering of measurements at arbitrary intervals. A traditional neural network struggles to handle this. Its design requires it to learn from data with clear stages of observation. Thus it is a poor

Thanks