Deep Learning and High Performance Computing Synergies

Bruno Raffin,
DataMove
INRIA, Univ. Grenoble Alpes

JIRC, Bourges, Fev 2020

Bruno.raffin@inria.fr

HPC-AI Synergies

- **AI-for-HCP:** Smart Infrastructure and resource management
- **HPC-for-AI**: Accelerating AI with HPC
- **AI-for-Science:** integration of numerical simulation and ML

Journées Convergence HPC-AI-Big Data, Nov 2019.

Slides from talks available at https://project.inria.fr/conv2019/

HPC versus BigData and ML

Parallelism for scalability

HPC

Performance comes first

Low level programming (MPI, OpenMP)

Thin software stack

Stable software libs

Tools developed by small communities

Target HPC centers

Jobs run a few hours on thousands of cores:

- Gysela (fusion):
  - 1 run = 10 M hours CPU
  - Scalable up to 0,5 M cores

Big Data and (shallow) ML

Ease of programming comes first

High level programming (Spark, Flink)

Thick software stack

Quickly changing software libs.

Tools developed by large communities

Target Cloud platforms

Jobs run a few days on tens of nodes:

- Spark deployed on 8000 nodes to process PBytes scale data

HPC versus BigData and ML

Parallelism for scalability

HPC

Performance comes first

Low level programming (MPI, Open

Thin software stack

Stable software libs

Tools developed by small co

Target HPC centers

Jobs run a few hours on thou

- Gysela (fusion):
  - 1 run = 10 M hours C
  - Scalable up to 0,5 M co

Deep Learning

High level programming environments supported by a large community (Tensorflow, PyTorch)

AlphaGo Zero training: 70 hours on 64 GPU workers and 19 CPU servers [Silver'17]

Big Data and (shallow) ML

f programming comes first

ogramming (Spark, Flink)

stack

software libs.

large communities

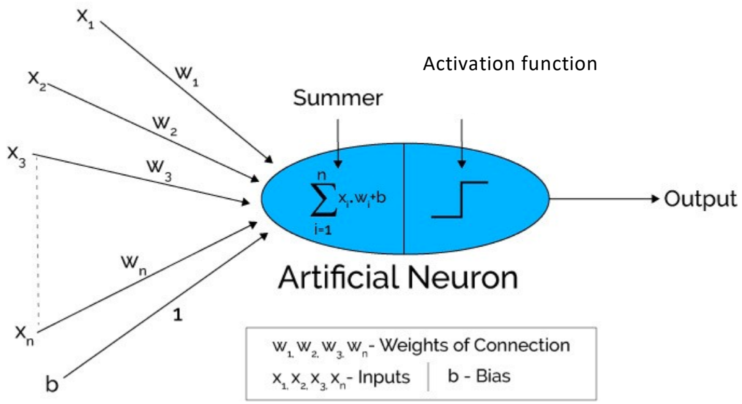
ms

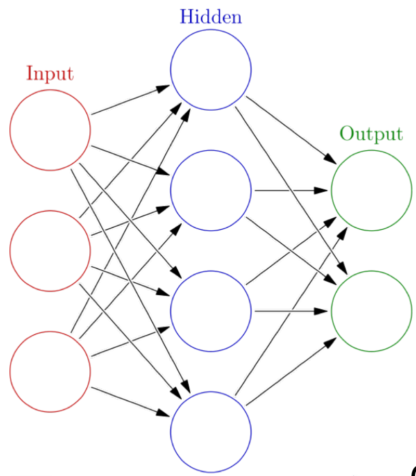
s on tens of nodes:

loyed on 8000 nodes to process

cale data

Artificial Neural Networks





Example x_i

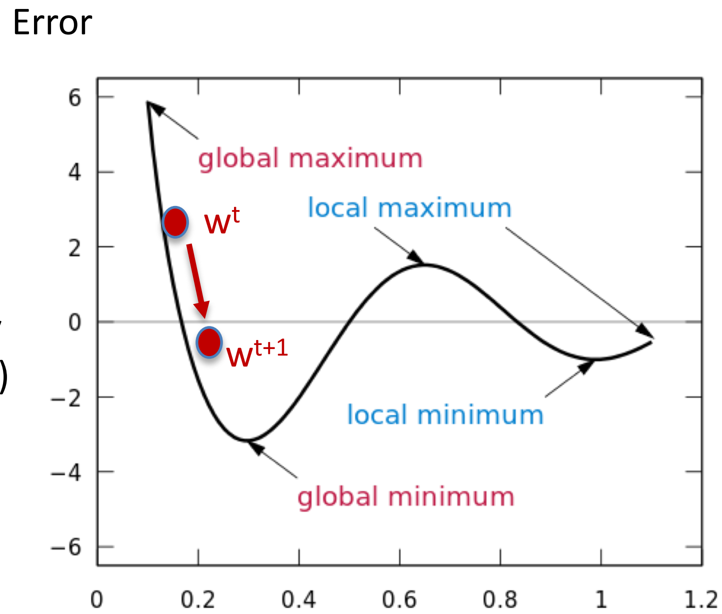
Output: y_i

Error $E(y_i, y'_i)$
(loss function)

Backpropagate error and compute weight updates

Usually examples are processed by batches

Parameter update (weighted by learning rate)

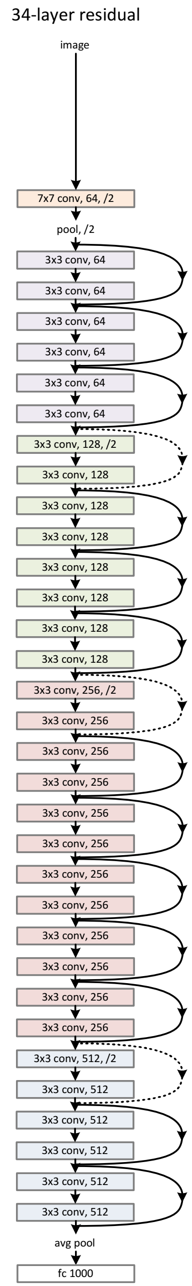


Parameter space

Backpropagation: weight optimization by stochastic Gradient descent

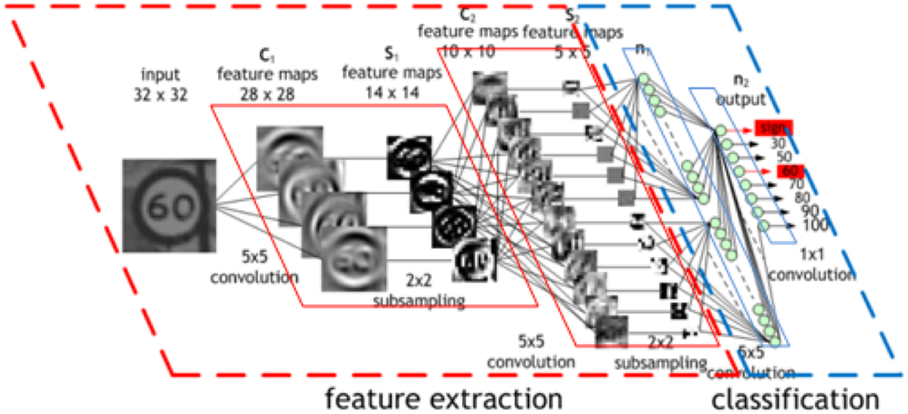
Deep Learning

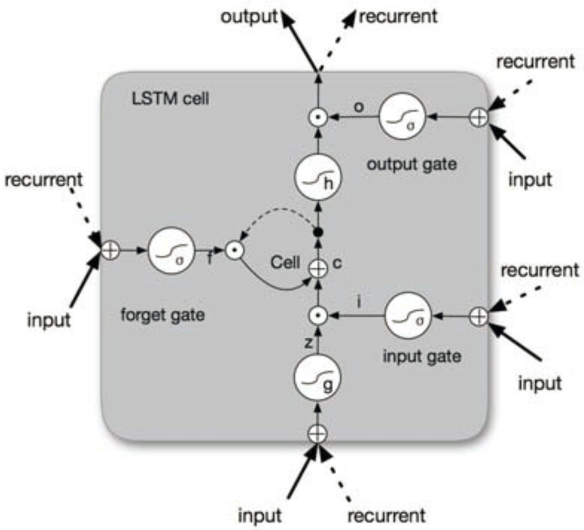
ResNet-34



Today’s neural networks are deep and complex:

Network zoology:
- MLP
- CNN
- Graph-CNN
- LSTM
- Attention NN
- ……





Megatron-LM [Shoeybi-19]:
- Architecture:
  72-layer, 8.3 billion parameters
- Training:
  174GB of text, 12 ZettaFLOPs, 9.2 days, 512 GPUs

The ResNet-50 Race

Table 1 : Training time and top-1 1-crop validation accuracy with ImageNet/ResNet-50

		Batch Size	**Processor**	**DL Library**	**Time**	**Accuracy**
2016	He et al. **[7]**	256	Tesla P100 x8	Caffe	29 hours	75.3%
2017	Goyal et al. **[1]**	8K	Tesla P100 x256	Caffe2	1 hour	76.3%
2017	Smith et al. **[4]**	8K→16K	full TPU Pod	TensorFlow	30 mins	76.1%
2017	Akiba et al. **[5]**	32K	Tesla P100 x1024	Chainer	15 mins	74.9%
2018	Jia et al. **[6]**	64K	Tesla P40 x2048	TensorFlow	6.6 mins	75.8%
2018	Mikami et al.	**34K→68K**	**Tesla V100 x2176**	**NNL**	**224 secs**	**75.03%**

A 28 000 x perf improvement in 3 years!

Parallelizing Deep Learning

Parameter update function

Generic learning process: $W^t = F(D, W^{t-1})$

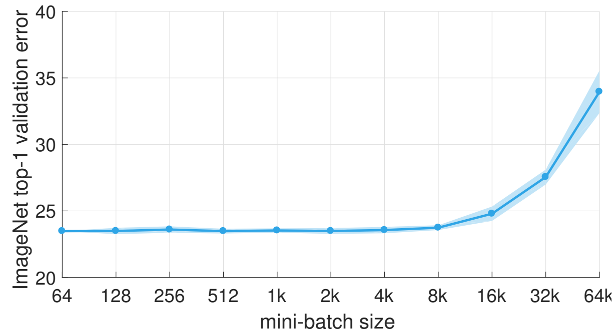
Learning Data

Model Parameters

Parameters updates are computed after presenting a batch of examples (batch learning)

2 main sources of parallelism:

- **Data parallelism**: distribute the learning set
- **Model parallelism**: distribute the model parameters

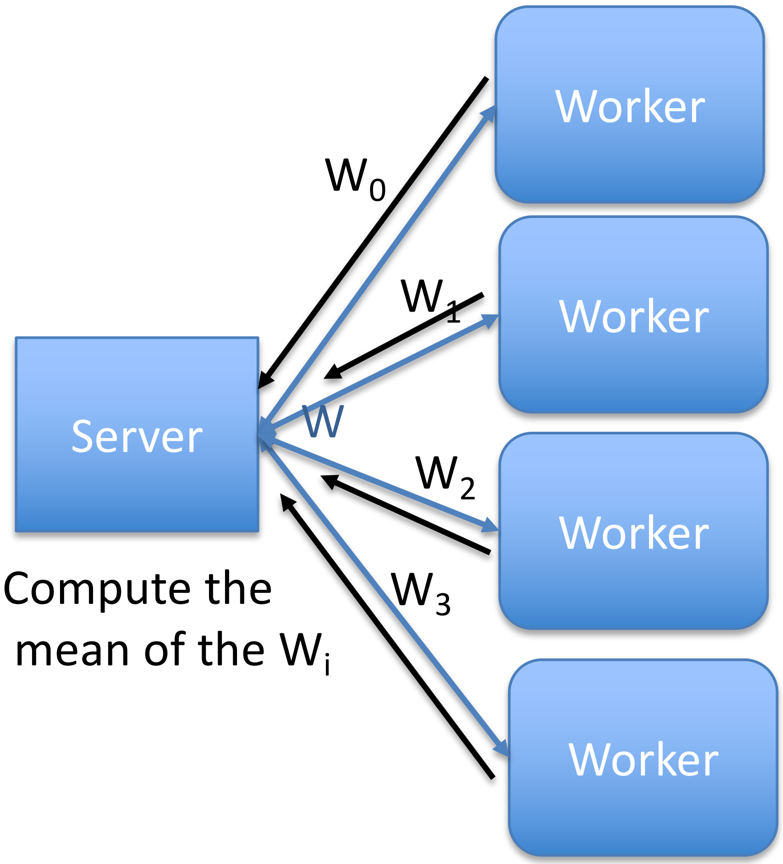


[Goyal 2017]

Data Parallelism

Duplicate the model (one per worker)

Partition the batch into P mini-batches, one per worker



Synchronous update (TensorFlow):

Loop:
Server sends parameters to all Workers;
Workers compute parameter updates on their mini-batch;
Server get updates from all Workers;
Server compute a global model update;
Server update parameters;
EndLoop

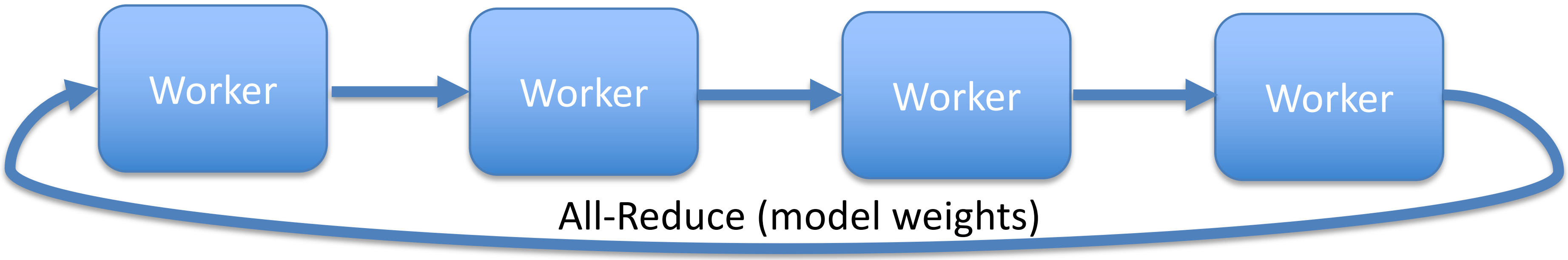
Limitations:

- Server is a bottleneck: gets P sets of model parameters

- Scaling the batch size affects the learning convergence

Data Parallelism

Fix the bottleneck: suppress the server and perform a all-reduce collective communication



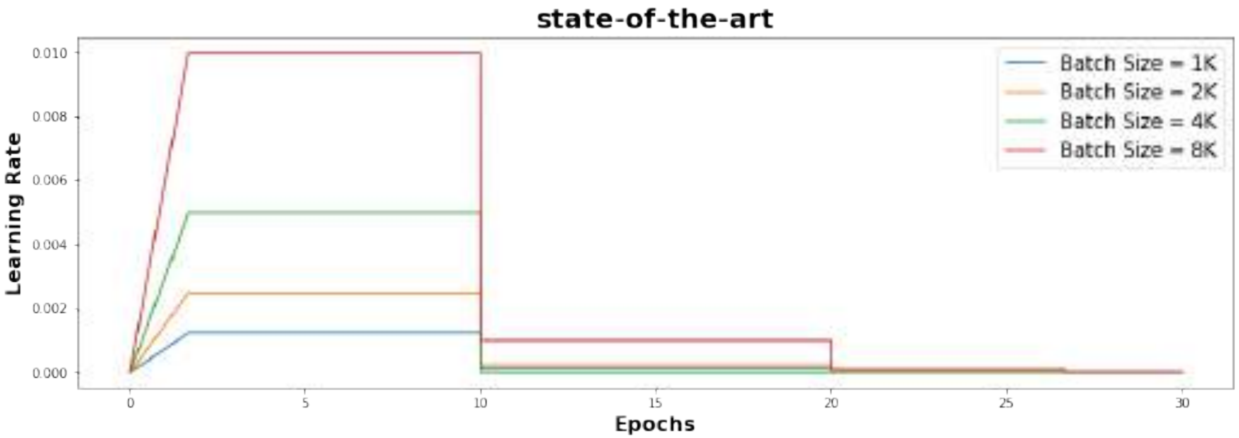
Communication cost per worker is now asymptotically independent on the number of workers

$$T(n) = 2\alpha \log(p) + 2\ n/\beta\ (p-1)/p + g\ .\ n\ .\ (p-1)/p$$

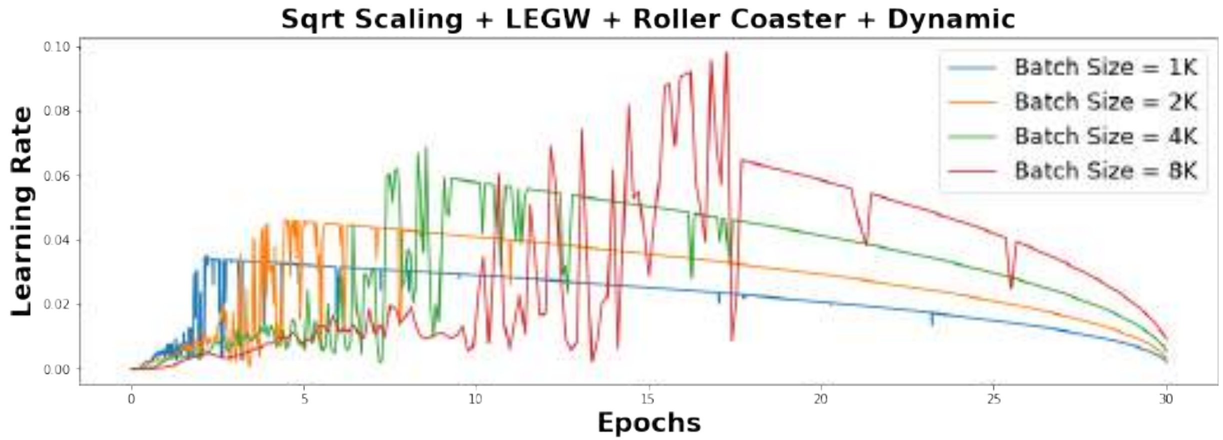
Strategy available using Horovod+Tensorflow (MPI based)

Data Parallelism

But scaling the batch size also requires adapted learning rate management to ensure a proper training convergence.



Warmup



You et al, SC 2019

Model Parallelism

Data parallelism limitation: not adapted if the neural network does not fit into memory

Model parallelism: Internal NN parallelization

- Difficult to acheive (tight data dependencies betwen neurons)
- 2 main approaches today:
  - Layer-wise pipe-lining (ex: Gpipe – Huang et al. 2018)
  - Distributed tensor computation (Mesh-TensorFlow – Shazeer et al. – 2018)

Data and Model parallelism can be combined (Megatron - 2019)

https://arxiv.org/abs/1909.08053

Learning From Simulations: Deep Fluid

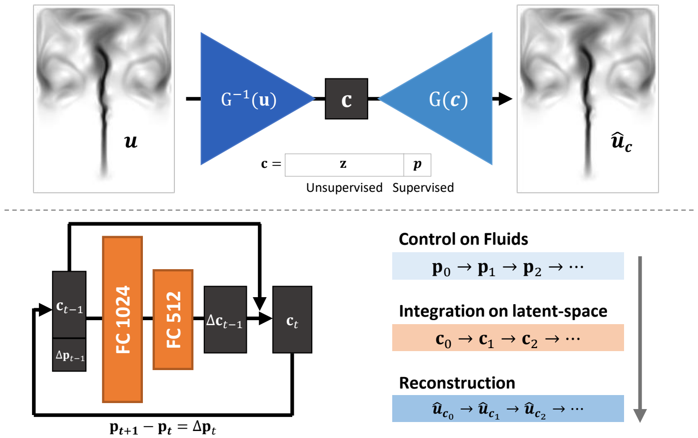
Simulation Output

Neural Network Output (on trained examples)

Train a neural network to reproduce the CFD simulation output from simulation data (varying the simulation input parameters)

Neural architecture:

Auto-encoder + Physics-inspired Loss



Kim et al. Eurographics 2019 https://arxiv.org/abs/1806.02071

Neural Surrogates for Massive Parametric Space Exploration

Taking benefit of the generalization capabilities of NN:

1) Use classical simulation to generate 10^5 simulation results from random combinations of parameter values.

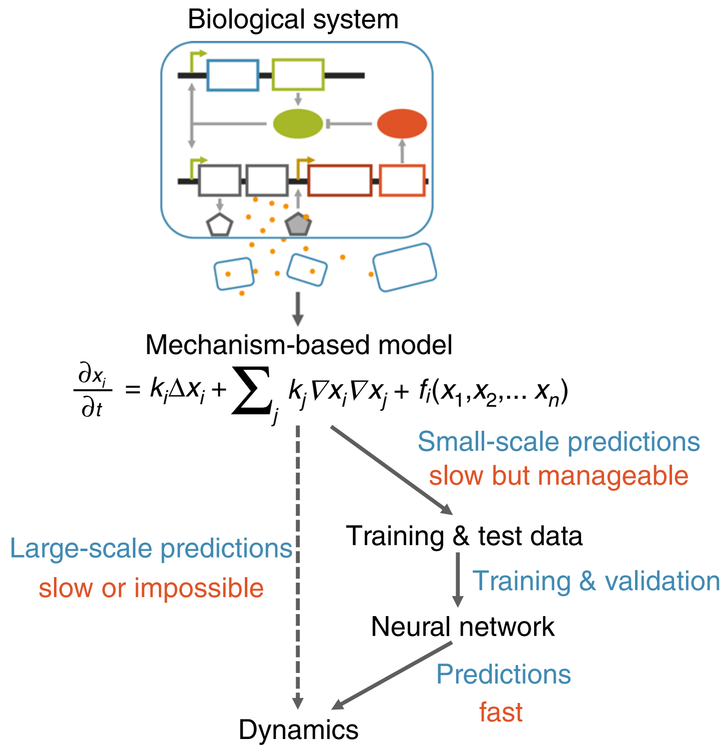
   2 months on a 400 node cluster

2) NN training from simulation data (LSTM arch)

3) Use NN to screen 10^8 combinations of parameter values

   12 days – instead of 986 year with standard simulation (NN 30 000x faster)

-> Found novel patterns



NN 30000x faster than sim.

Wang et al. 2019, Nature Communication

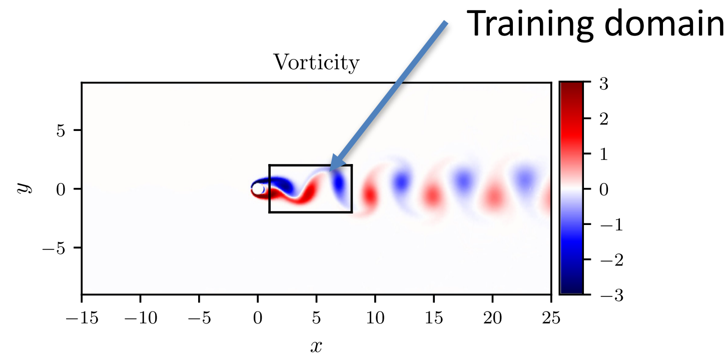
Physics-Inspired Neural Networks

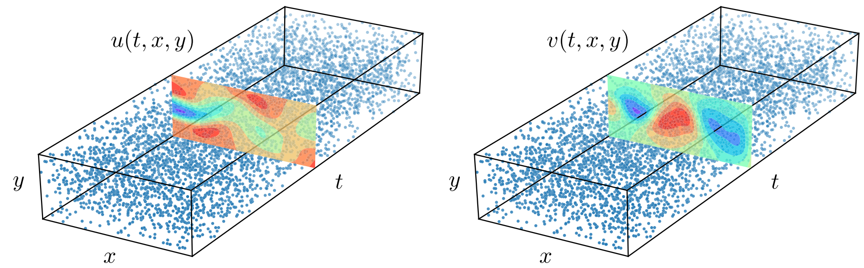
2D Navier-Stokes equations:

$$u_t + \lambda_1 (u u_x + v u_y) = -p_x + \lambda_2 (u_{xx} + u_{yy}),$$
$$v_t + \lambda_1 (u v_x + v v_y) = -p_y + \lambda_2 (v_{xx} + v_{yy}),$$

u,v: velocity fields

p: pressure





Neural network with loss function

Enforcing physics constraints:

Classical loss **Navier-Stokes**

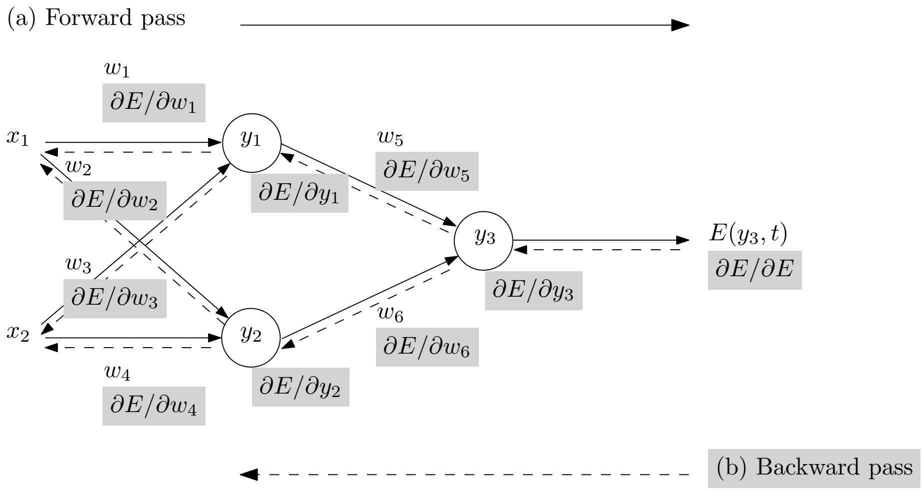
$$MSE := \frac{1}{N}\sum_{i=1}^{N}\left(|u(t^i, x^i, y^i) - u^i|^2 + |v(t^i, x^i, y^i) - v^i|^2\right) + \frac{1}{N}\sum_{i=1}^{N}\left(|f(t^i, x^i, y^i)|^2 + |g(t^i, x^i, y^i)|^2\right).$$

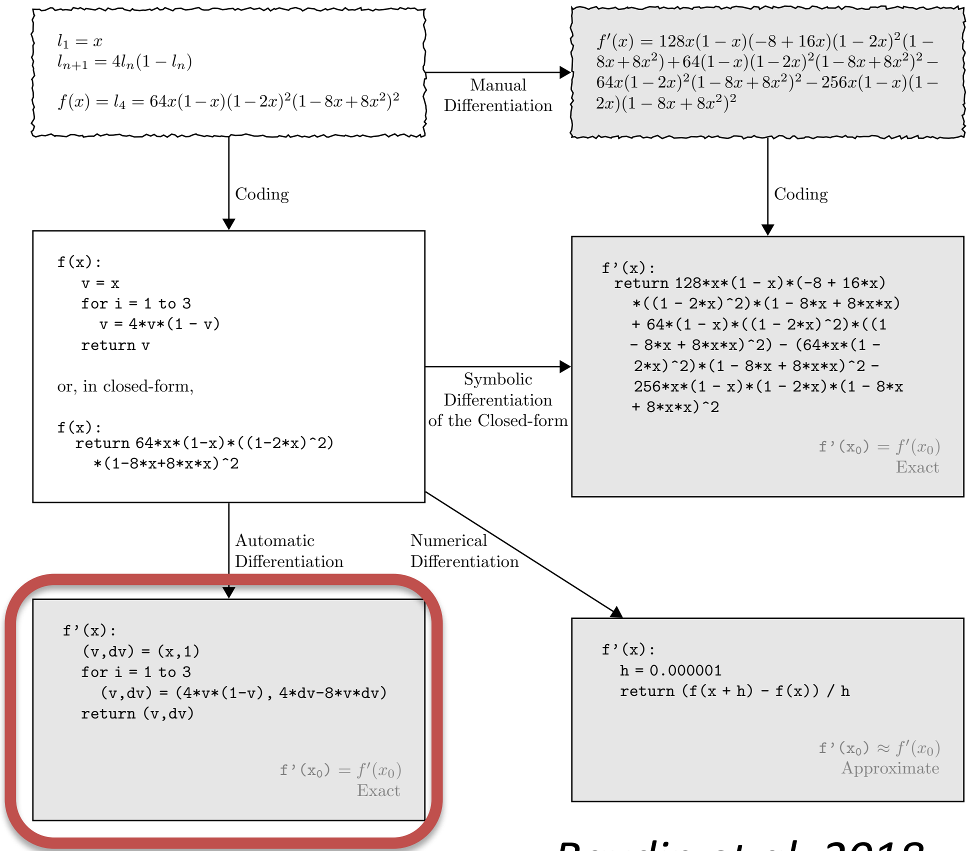
$$f := u_t + \lambda_1 (u u_x + v u_y) + p_x - \lambda_2 (u_{xx} + u_{yy}),$$
$$g := v_t + \lambda_1 (u v_x + v v_y) + p_y - \lambda_2 (v_{xx} + v_{yy}),$$

Learn pressure field, Delta 1 and Delta 2

Raissi et al, JoCP 2018

Differentiable Programming





Baydin et al. 2018

Automatic differentiation:

- Initially developed to produce adjoint code automatically
- Generalized backpropagation algorithm
- Supported by standard libs like PyTorch and TensorFlow.

From (static) neural network programming to differential programming (**dynamic deep architectures**).

Myia language

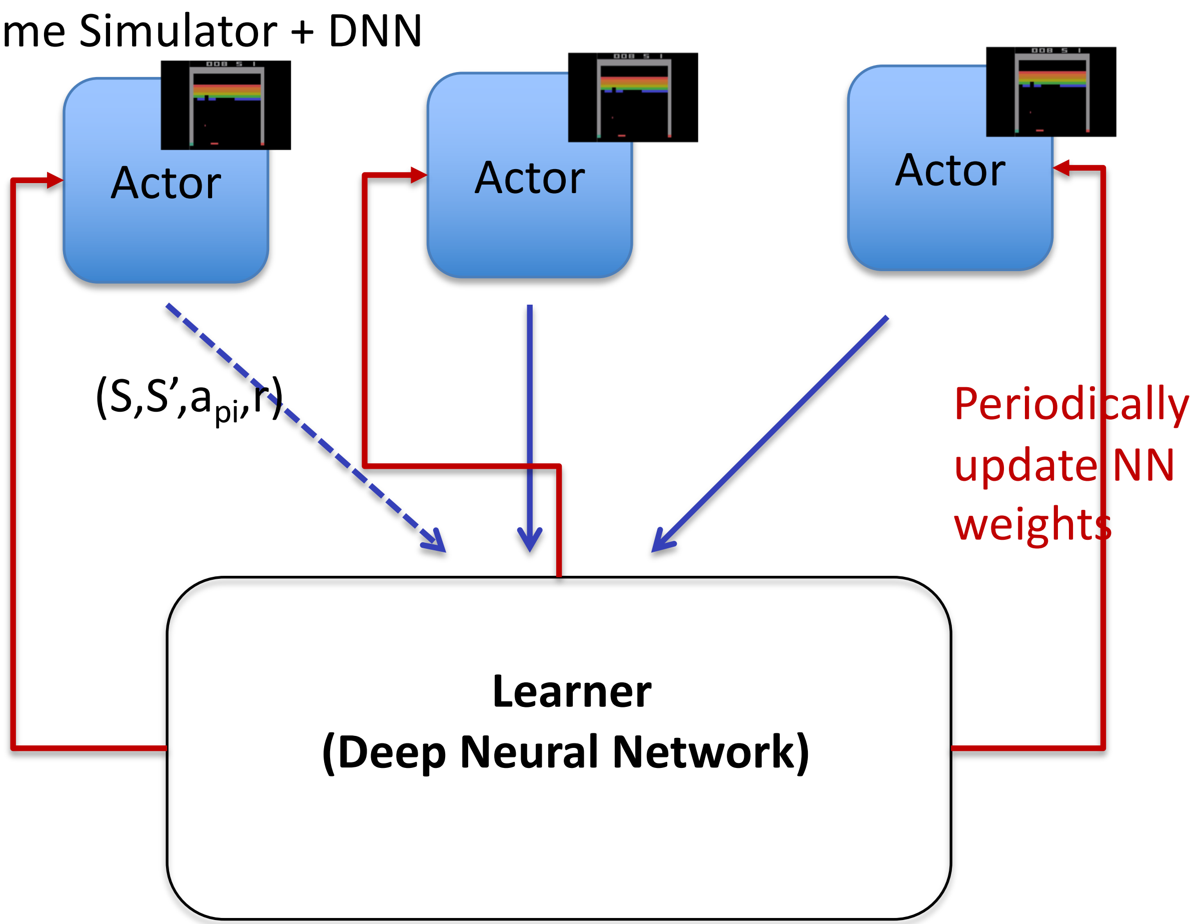
Memory, automatic differentiation and checkpointing:

- Store versus recompute intermediate states required for the backward pass
- Use checkpointing approaches to control the amount of memory required

Beaumont et al, 2019, PTRSA.

Deep Reinforcement Learning

Actor-Critic Schematic Architecture



(S, S', a_{pi}, r):

- S & S’: states
- Pi: Policy
- a_{pi} : action taken according to policy Pi.
- r: reward

Some recent strategies for DRL: A3C, Impala
Framework for DRL: Ray Lib

AlphaGo Zero: trained during more than 70 hours using 64 GPU workers and 19 CPU parameter servers *[D. Silver, Nature 2017]*

Auto Deep Learning

Hyperparameters:

- Every other NN architecture parameter not computed by the backpropagation
- Today hyperparam. setting is mainly expert based
- AutoML: towards automatic hyperparameter setting

AutoML main approaches:

- Reinforcement Learning
- Genetic algorithms

See the RayTune lib for instance

And of course very compute intensive embarrassingly parallel workload (and people start to question carbon impact)

https://autodl.chalearn.org/

Conclusion

- HPC-for-AI:
  - Massive parallelism not yet common (but quickly changing domain). Day long trainings are commons. Design/test/share/reproduce (MLFlow)
  - Accelerators needed - GPUs (gammer cards do the job), TPUs….
  - Complexity is growing (NN architectures as well as NN assemblies like GAN, DRL).
  - AutoML (with transfert learning): towards ML factories ?

- AI-for-Science: a way to bring closer data and simulation
  - Massive data analytics
  - Integration of observation data into numerical simulations (data assimilation)
  - Physics-Inspired NN: ODE/PDE integration with NN + automatic differentiation

Reference Books

A classic about deep learning:

Deep Neural Networks, Ian Goodfellow and Yoshua Bengio and Aaron Courville

A classical about machine learning:

Pattern Recognition and Machine Learning, Christopher Bishop