Optimised Deep Learning - Jean Zay

Practical work (of your choice !)

cnrs
Institut du
Développement et des
Ressources en
Informatique
Scientifique

Like you want !

JIT ◀

Data Parallelism under the hood ◀

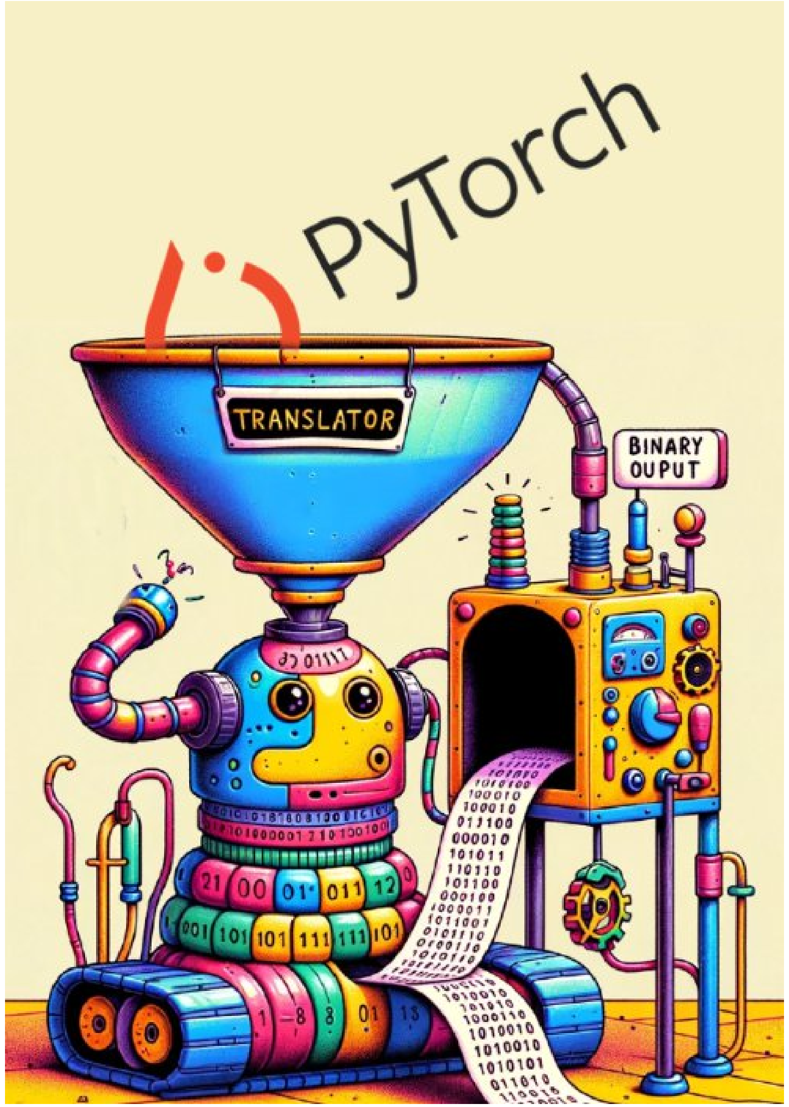
HPO ◀

Data Augmentation ◀

Large model ◀

Just-In Time Compilation

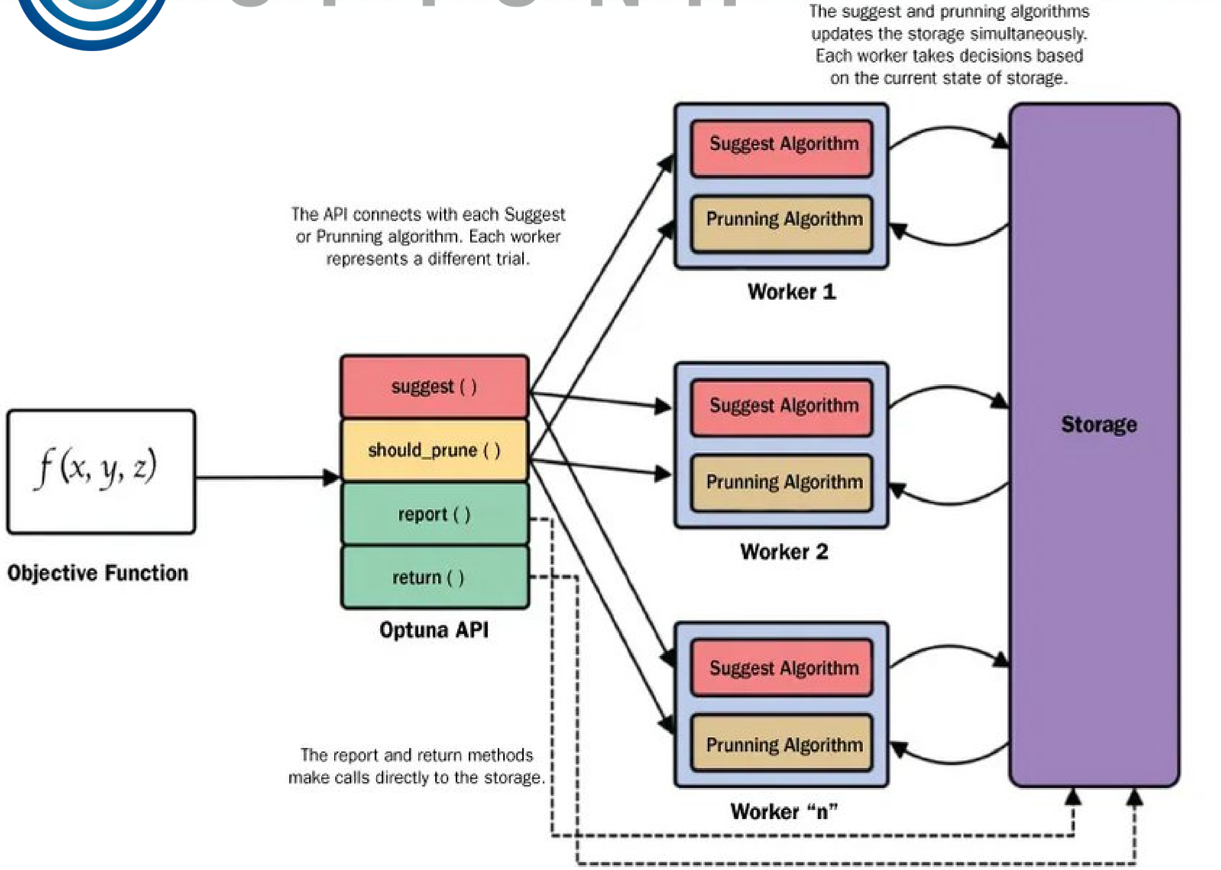
How to accelerate deep learning codes using *old school* compilation ?



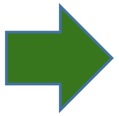
HPO at Scale

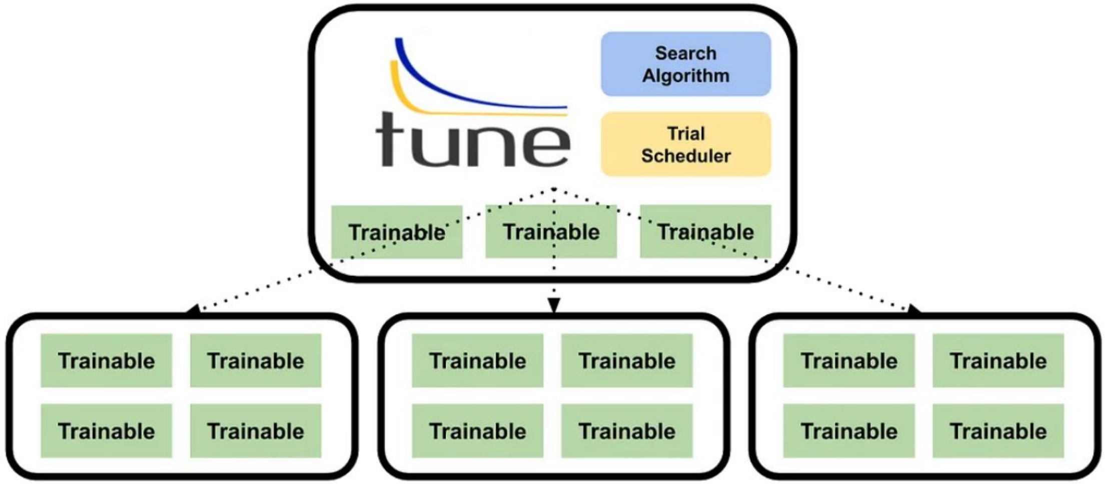
How to scale your HPO on supercomputer ?
How to use Optuna & Ray at scale ?

OPTUNA



The suggest and prunning algorithms updates the storage simultaneously. Each worker takes decisions based on the current state of storage.
The API connects with each Suggest or Prunning algorithm. Each worker represents a different trial.
f(x, y, z)
Objective Function
suggest ( )
should_prune ( )
report ( )
return ( )
Optuna API
Suggest Algorithm
Prunning Algorithm
Worker 1
Worker 2
Worker "n"
Storage
The report and return methods make calls directly to the storage.

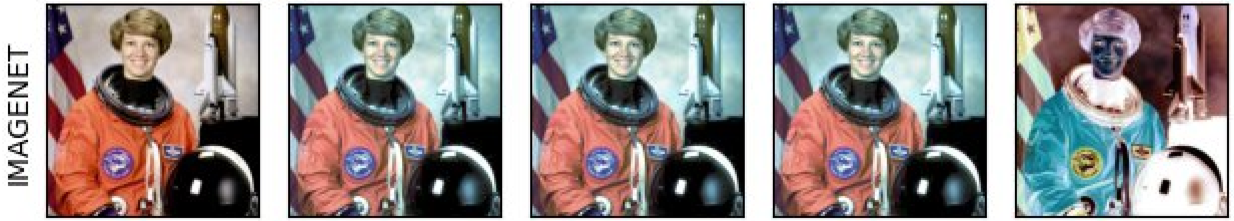




Data Augmentation

Is CPU enough?
How to delegate the Data Augmentation to the GPU?
How to optimize the Data Augmentation on the GPU?

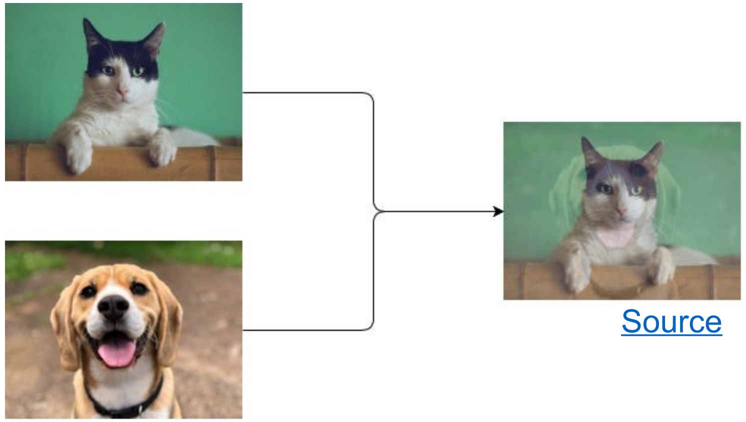
- **RandAugment** (torchvision): combinations of multiple transforms, either geometric or photometric, or both



Source

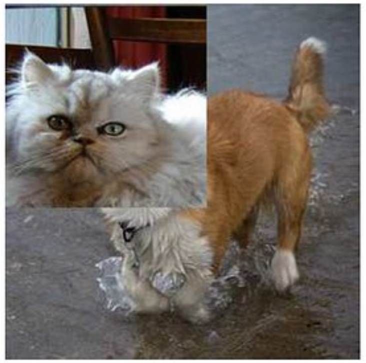
- **MixUp** (*mixup.py*): mixing up the features and their corresponding labels

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$



Source

- **CutMix** (*cutmix.py*): replacing an image region with a patch from another training image



Source

Large model

Pipeline Parallelism

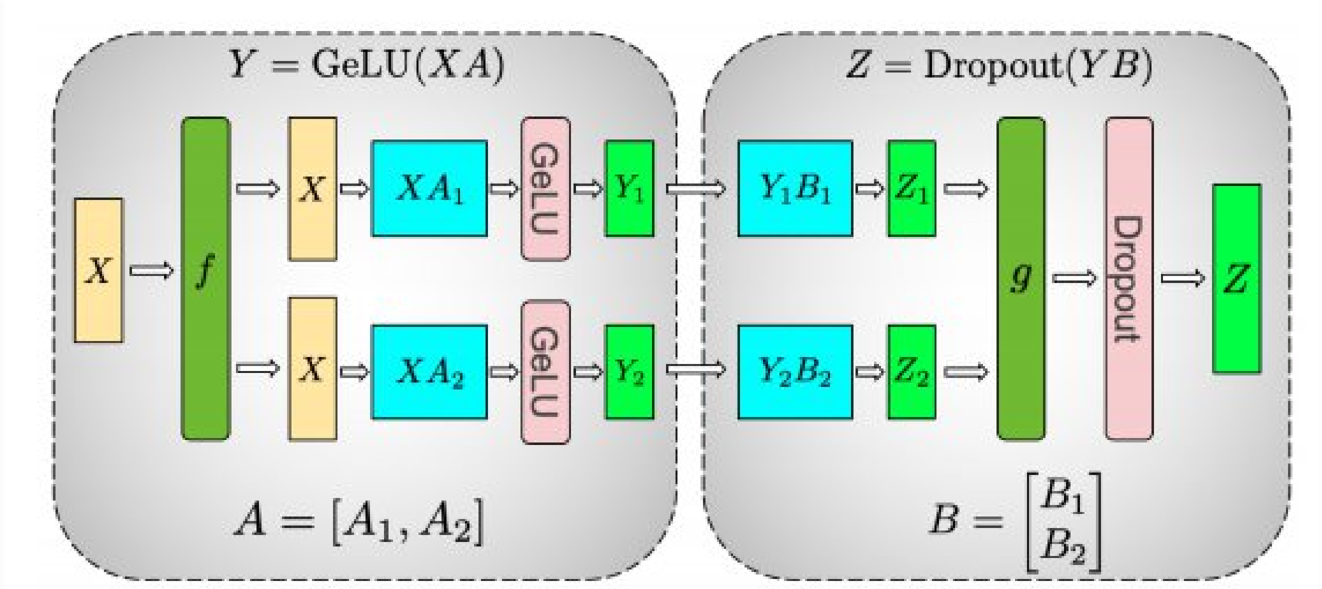
PyTorch

Coatnet-7
2440M parameters

deepspeed

Optimized Data Parallelism
ZeRO - FSDP

Tensor Parallelism under the hood



Conclusions

JIT ◄

Data Parallelism under the hood ◄

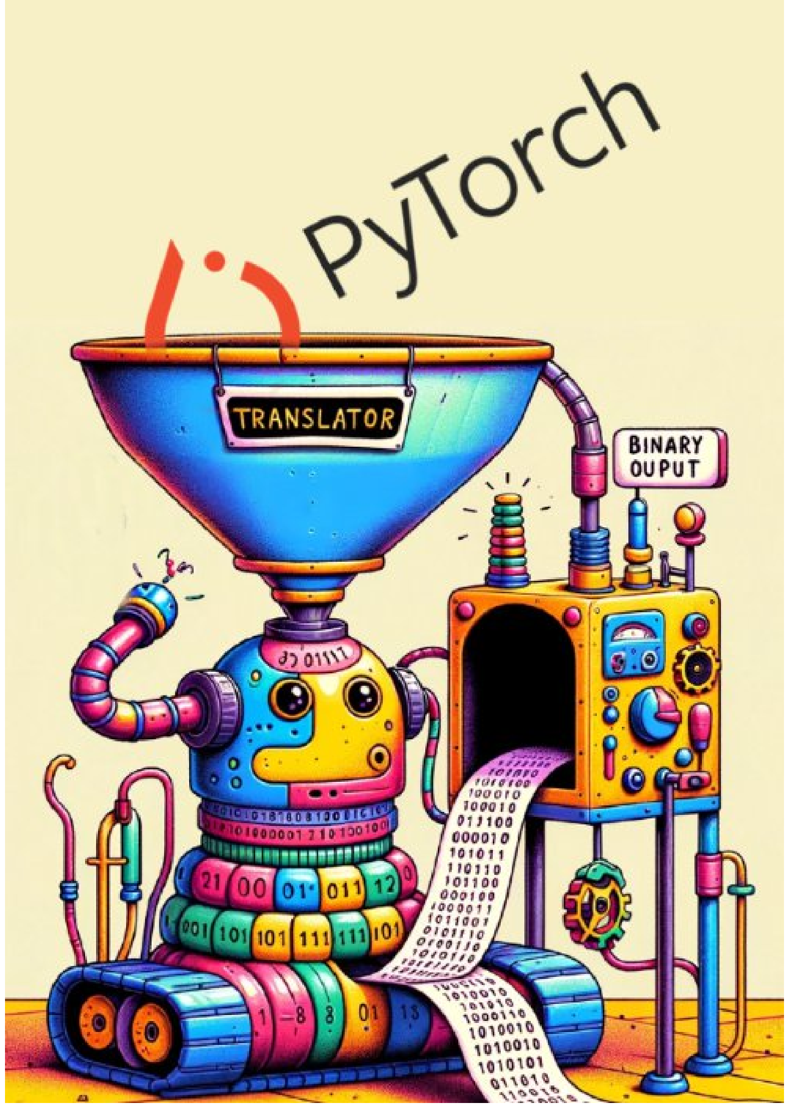
HPO ◄

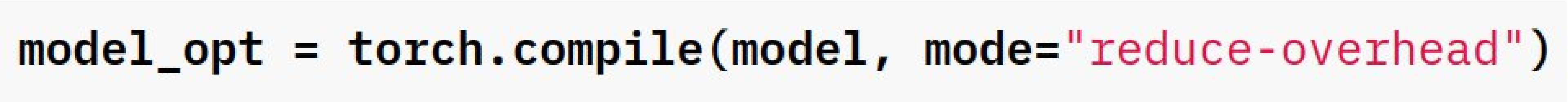
Data Augmentation ◄

Large model ◄

Just-In Time Compilation

How to accelerate deep learning codes using *old school* compilation ?





```
model_opt = torch.compile(model, mode="reduce-overhead")
```

HPO at Scale

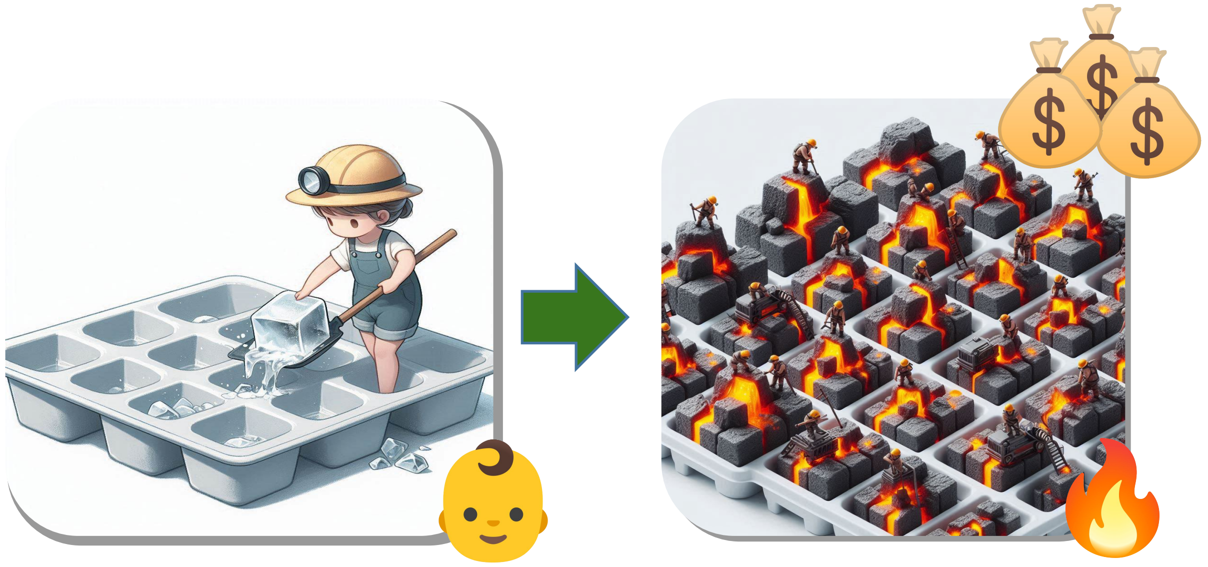
How to scale your HPO on supercomputer ?
How to use Optuna & Ray at scale ?

OPTUNA

- Database approach
- Not Multiprocess Native

RAY tune

- Multi-worker native
- Many parallel algorithms
- Difficult to use (especially with slurm)



Data Augmentation

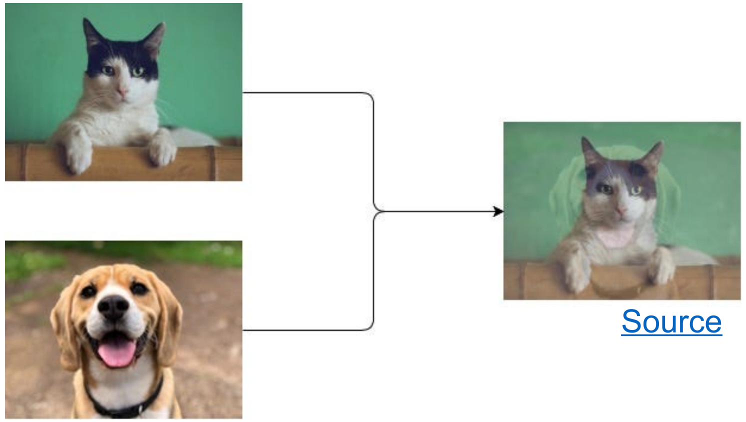
Is CPU enough?
How to delegate the Data Augmentation to the GPU?
How to optimize the Data Augmentation on the GPU?

- **RandAugment** (torchvision): **CPU is enough**

Source

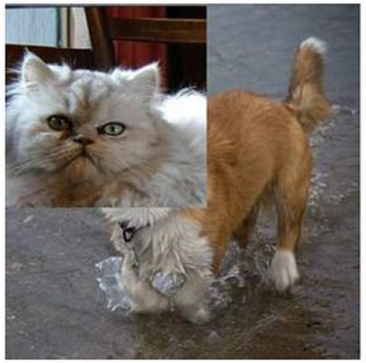
- **MixUp** (*mixup.py*): **much better on GPU**

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j, \qquad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j, \qquad \text{where } y_i, y_j \text{ are one-hot label encodings}$$



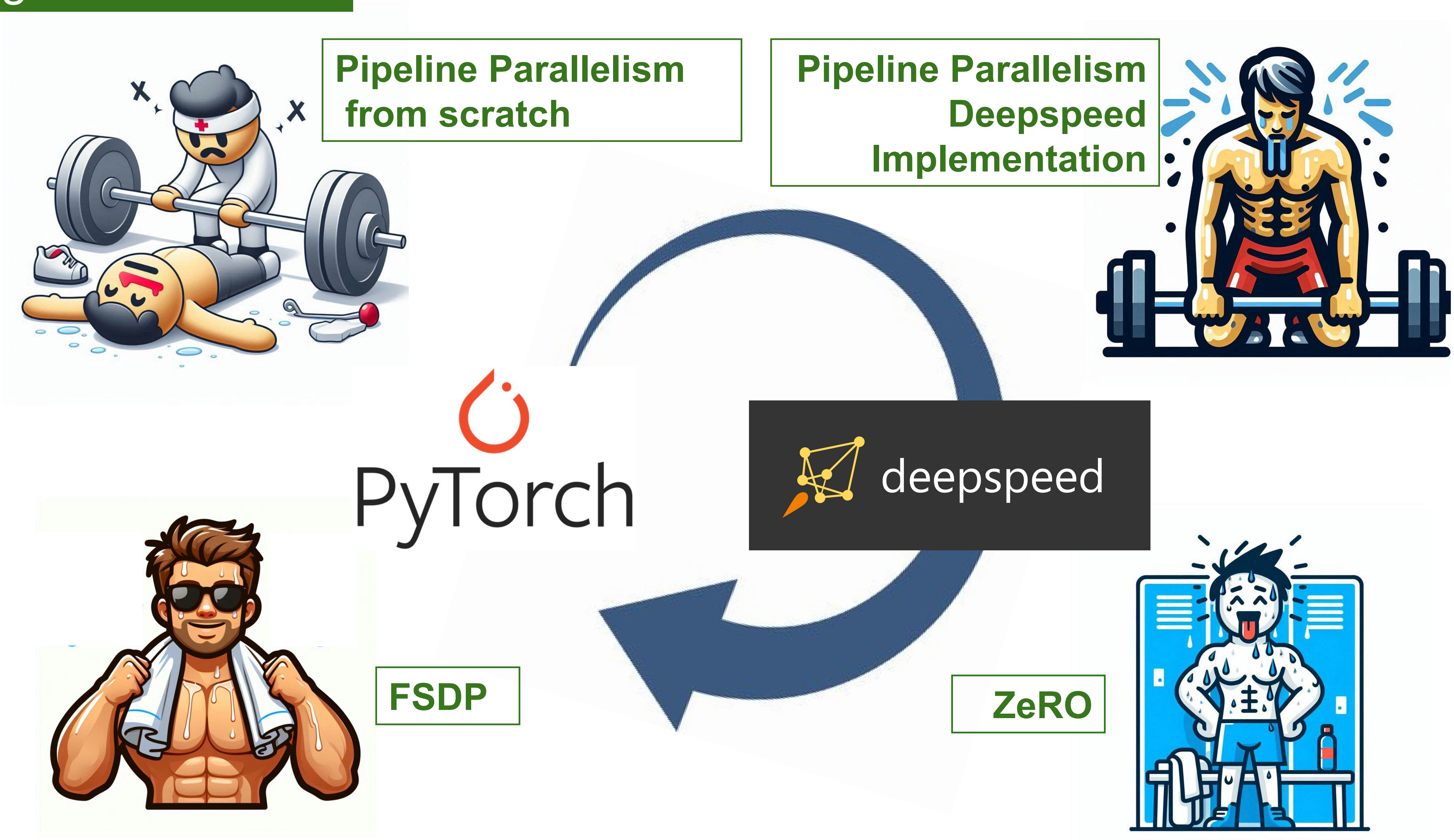
Source

- **CutMix** (*cutmix.py*): **tranform loop over images into loop over batches to improve parallelism and benifit from the GPU acceleration**



Source

Large model



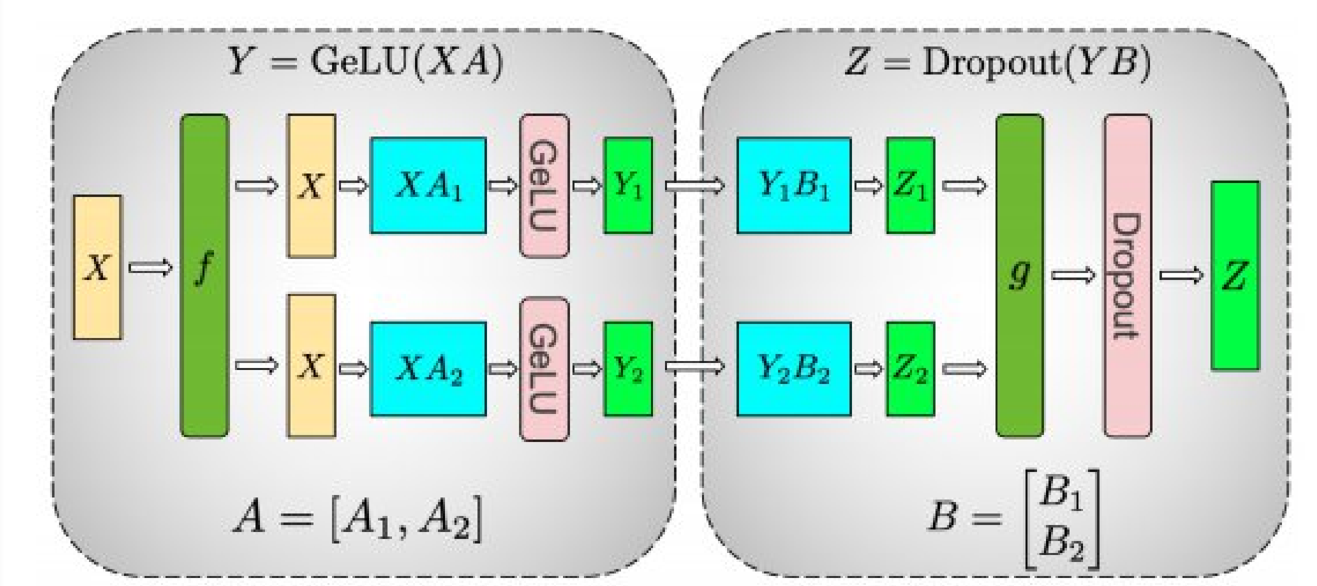
Pipeline Parallelism from scratch

Pipeline Parallelism Deepspeed Implementation

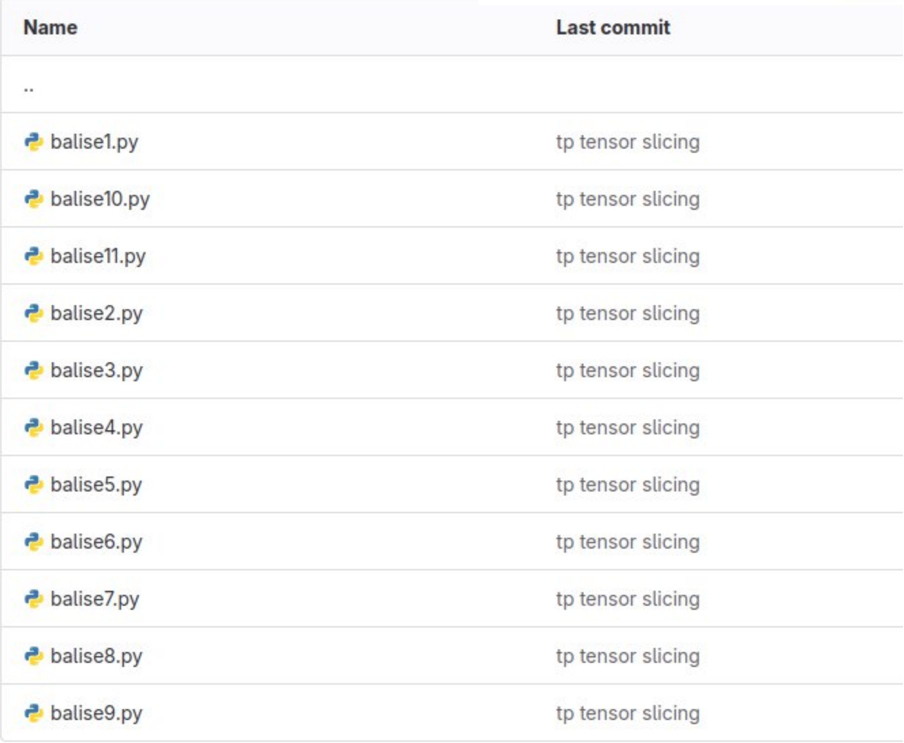
FSDP

ZeRO

Tensor Parallelism under the hood



tp tensor slicing
Nathan CASSEREAU authored 3 months ago



Name	Last commit
..	
balise1.py	tp tensor slicing
balise10.py	tp tensor slicing
balise11.py	tp tensor slicing
balise2.py	tp tensor slicing
balise3.py	tp tensor slicing
balise4.py	tp tensor slicing
balise5.py	tp tensor slicing
balise6.py	tp tensor slicing
balise7.py	tp tensor slicing
balise8.py	tp tensor slicing
balise9.py	tp tensor slicing