Deep Learning Optimisé - Jean Zay

Les parallélismes des gros modèles

INSTITUT DU
DÉVELOPPEMENT ET DES
RESSOURCES EN
INFORMATIQUE
SCIENTIFIQUE

Les Parallélismes de modèle pour les très gros modèles

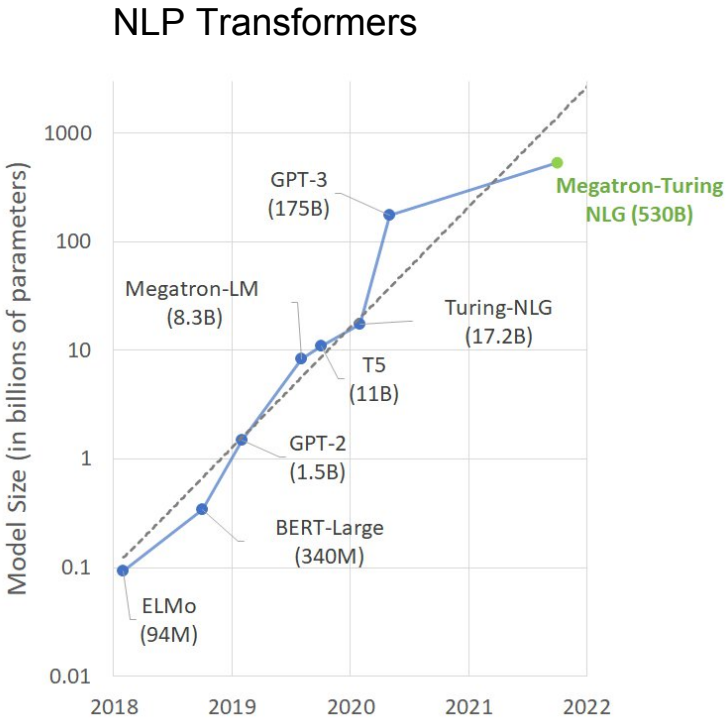
Pipeline parallelism ◂

Tensor parallelism ◂

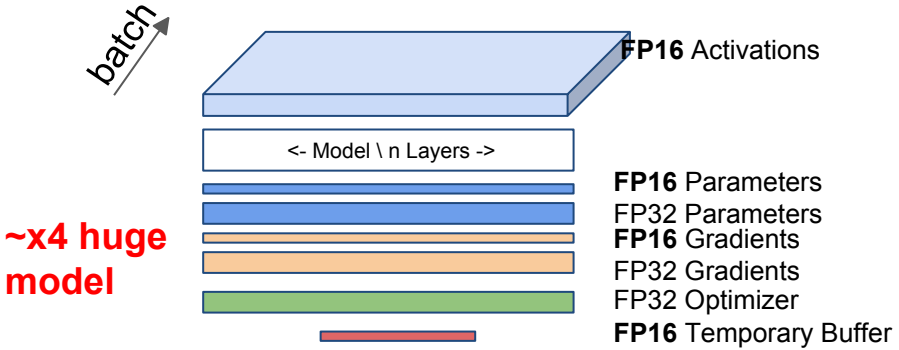
Hybrid parallelism ◂

3D parallelism ◂

Énormes modèles > 1 Milliard de paramètres



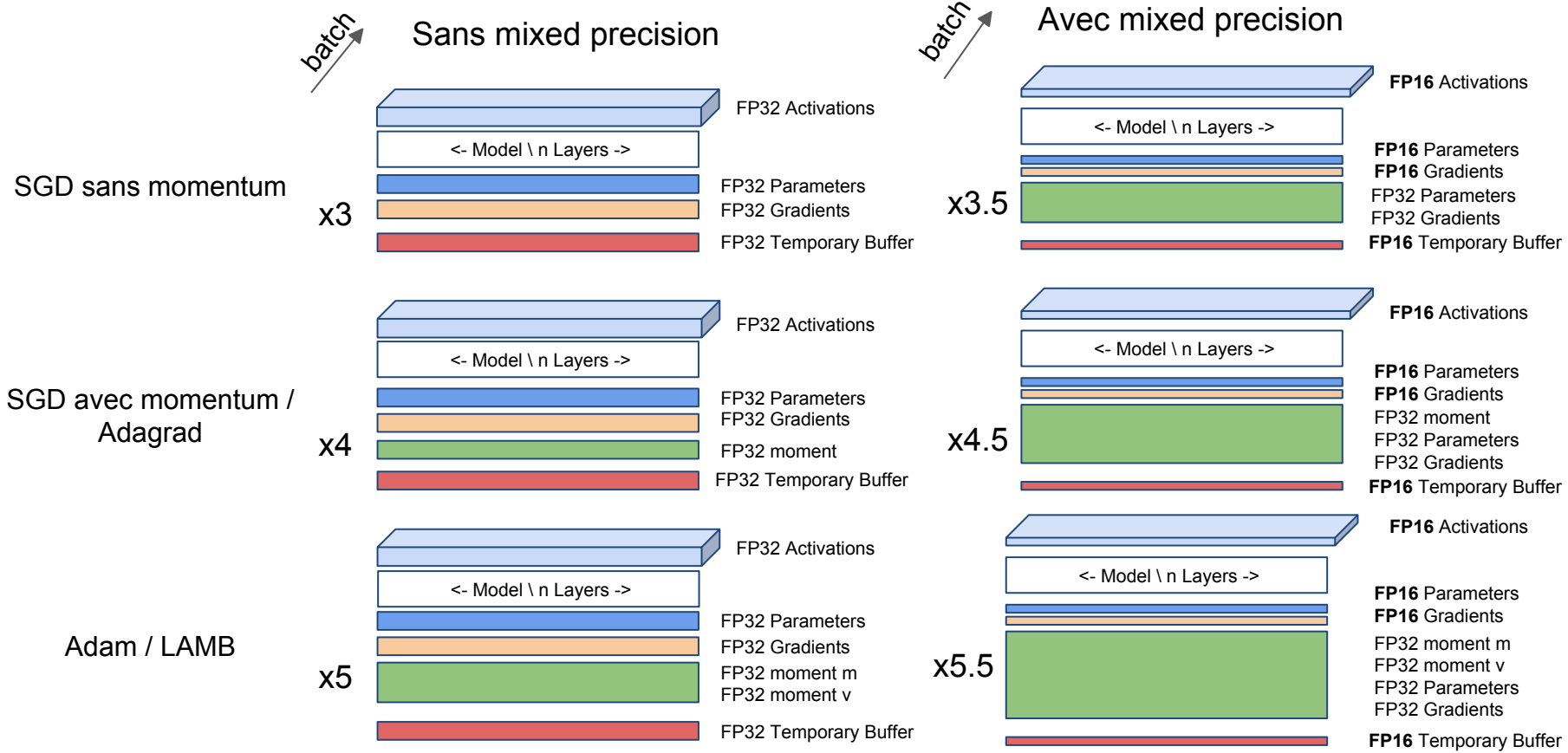
CUDA Out Of Memory



~x4 huge model

[1]

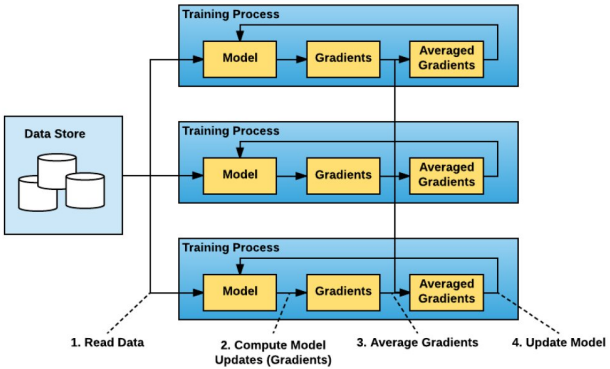
Empreinte mémoire

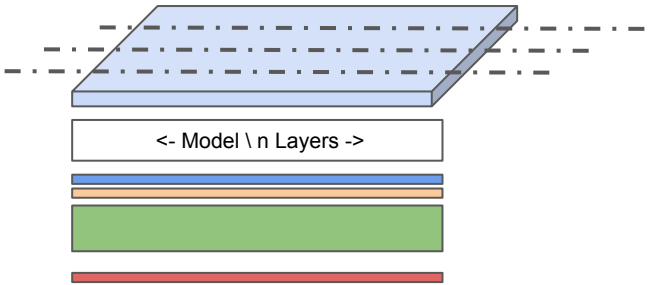


Les différents parallélismes

Data Parallelism

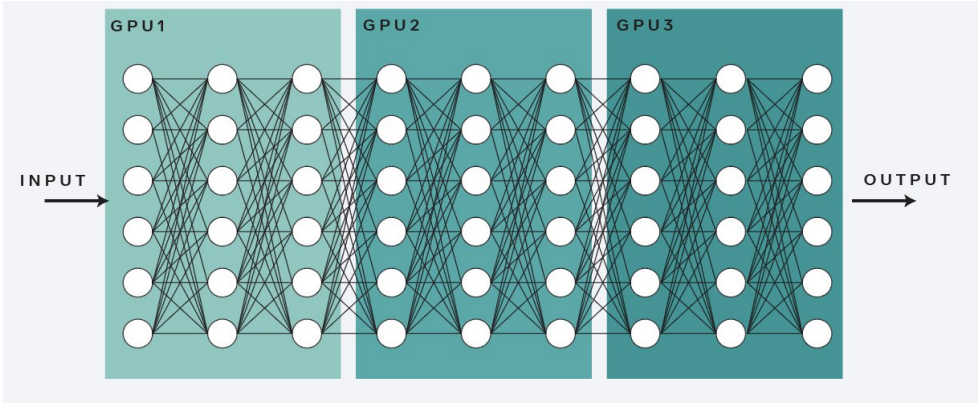
- Meilleur Throughput
- Seule l'empreinte mémoire des activations est distribuée
- Multi Processing

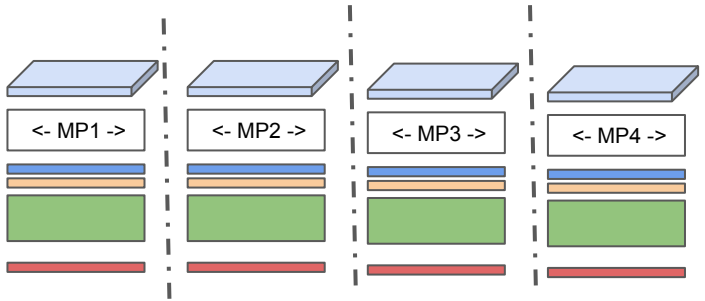




Pipeline Model Parallelism

- Empreinte mémoire distribuée
- Mono ou multi-processing

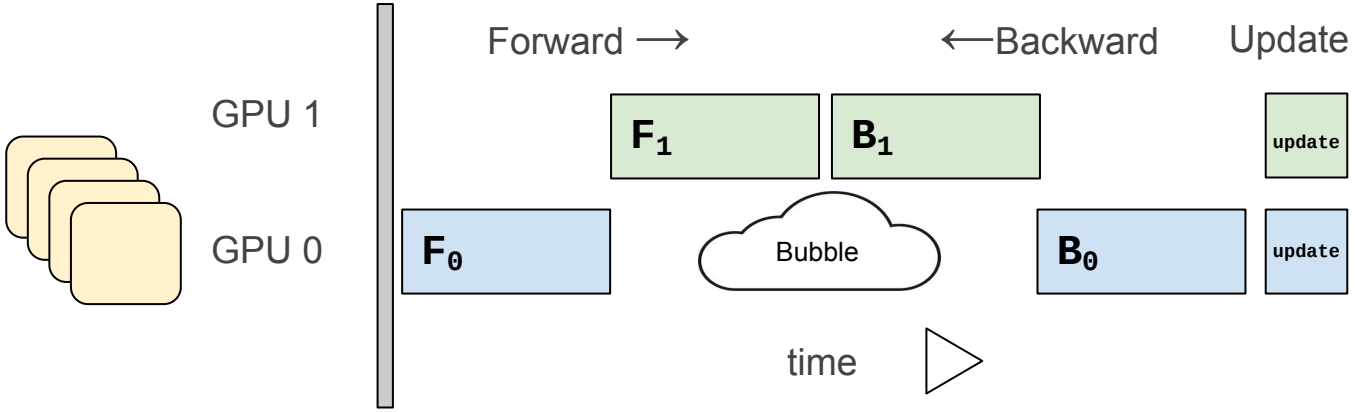


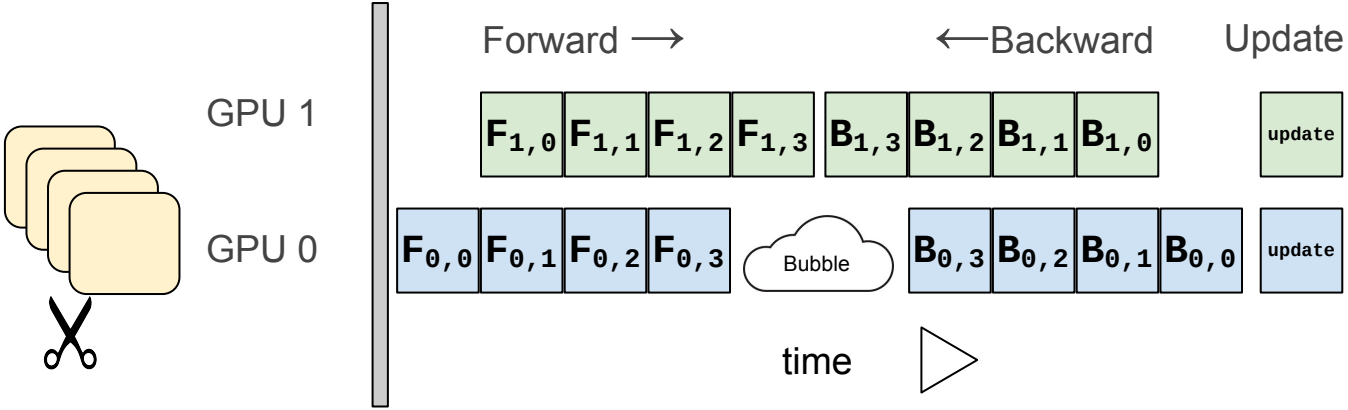


[2, 3]

Pipeline Parallelism

Model parallelism naïf sur 2 GPU



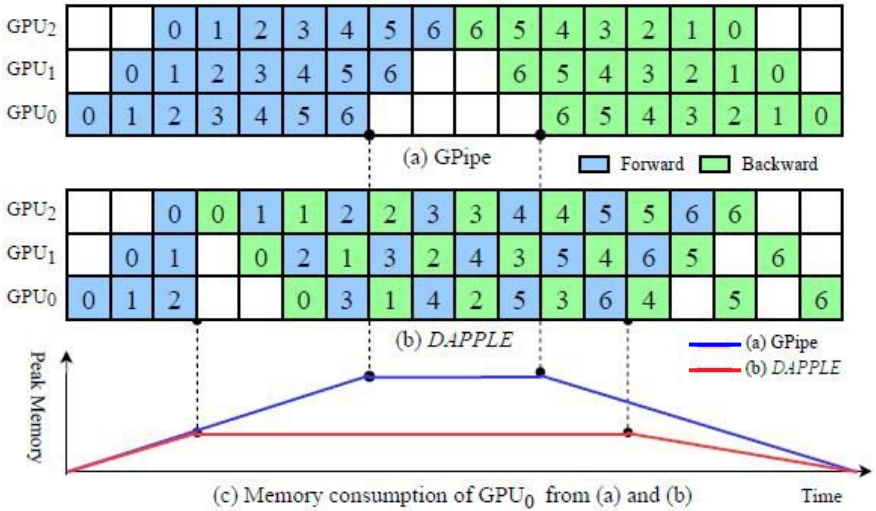


Model parallelism sur 2 GPU en **pipeline**

Pipeline Parallelism

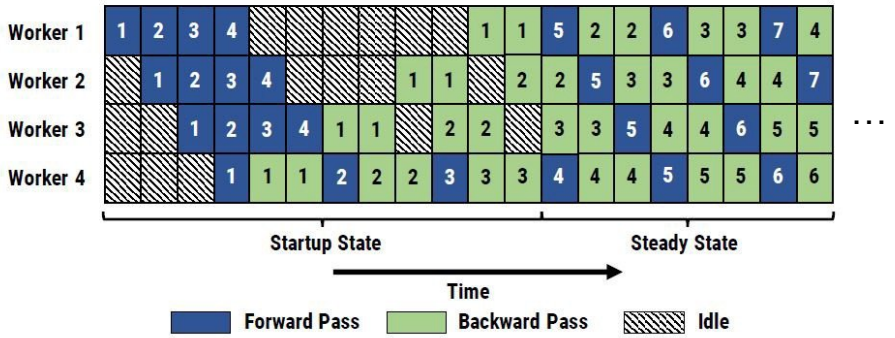
Synchronous pipeline :
GPipe, DAPPLE

- \- Throughput
- \+ Memory consumption
- \+ Convergence



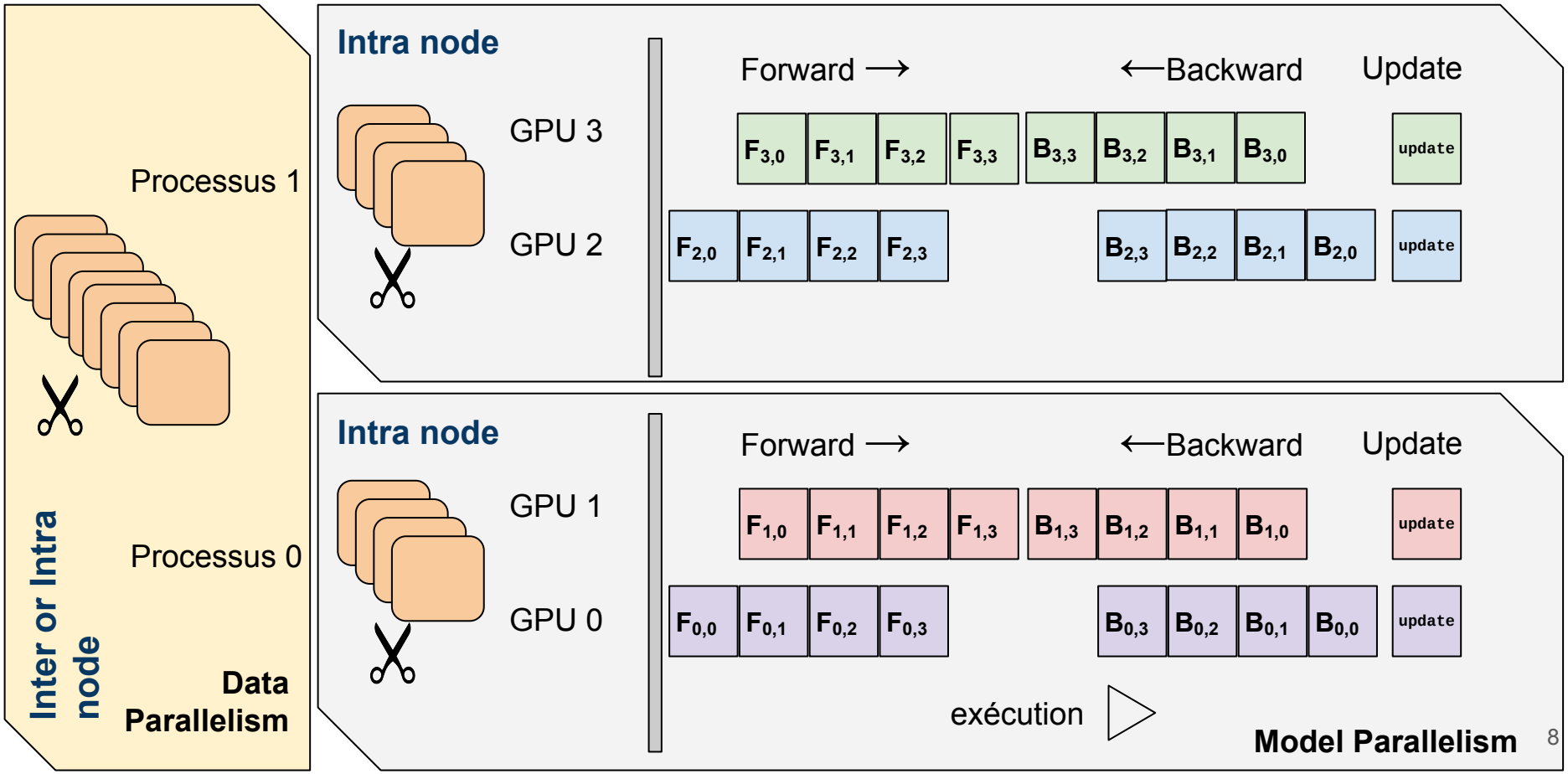
Asynchronous pipeline :
PipeDream, PipeMare

- \+ Throughput
- \- Memory consumption
- \- Convergence

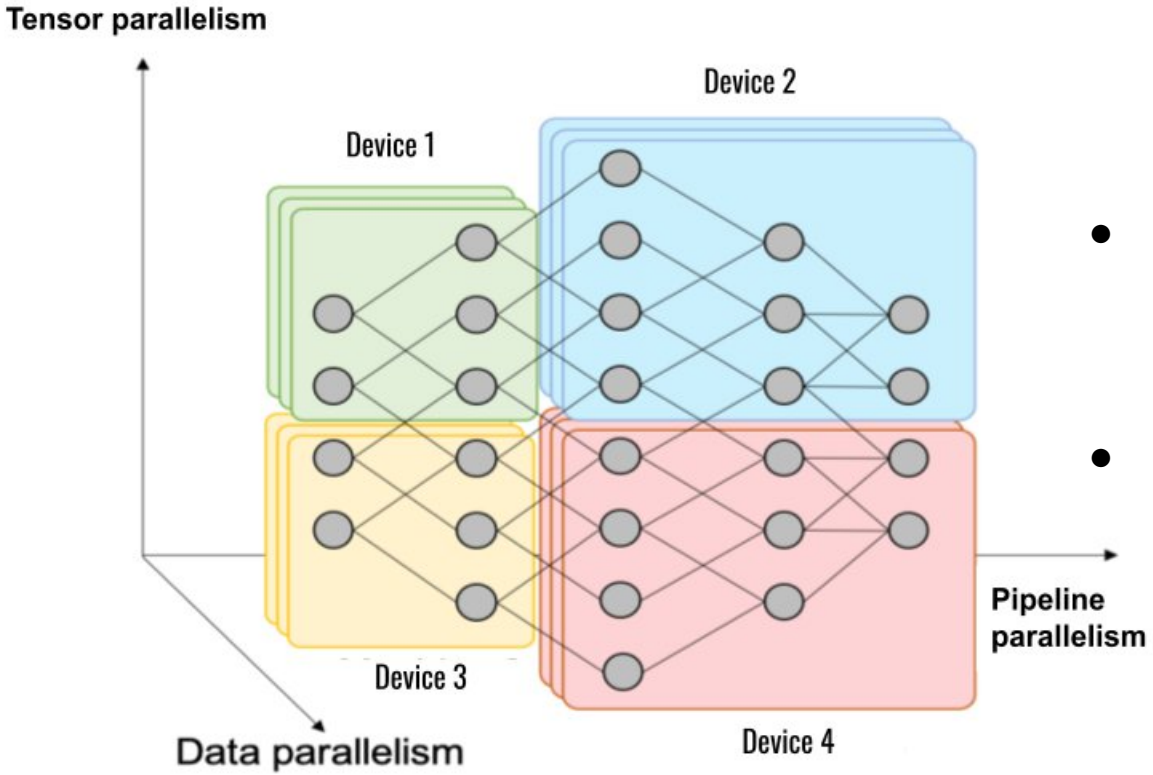


[4, 5]

Hybrid Parallelism



3D Parallelism



- **Data Parallelism**
  - Simple à implémenter
  - Meilleure performance
  - Augmente la taille du batch (problème de convergence)
- **Pipeline Parallelism**
  - Effort d'implémentation.
  - Équilibre entre mémoire, performance et convergence.
- **Tensor Parallelism**
  - Effort important d'implémentation
  - Bonne accélération des calculs
  - **Bande passante très sollicitée** (implémentation Intra-nœud)

API parallélismes de modèle

Deepspeed ◂
Colossal-AI ◂
Megatron-LM ◂

Deepspeed

Model Scale
Support 200B
Toward 100 Trillion

Speed
Up to 10x faster

Scalability
Superlinear speedup

Usability
Few lines of code changes

```
# Include DeepSpeed configuration arguments
parser = deepspeed.add_config_arguments(parser)
```

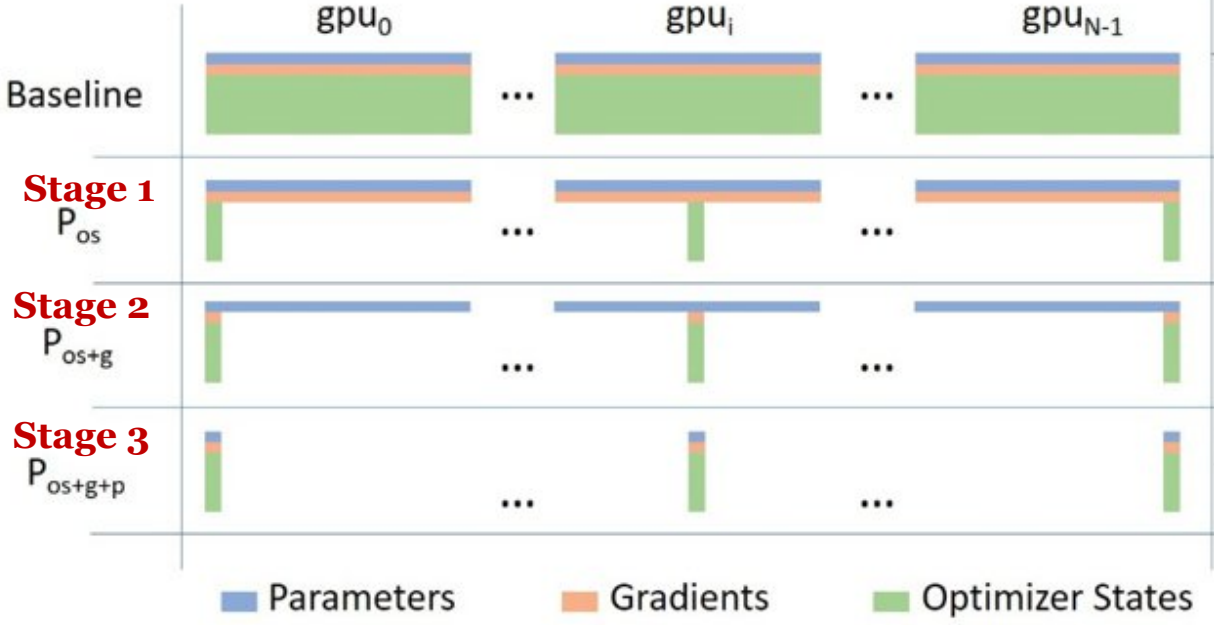
```
# Initialize DeepSpeed to use the following features
# 1) Distributed model
# 2) DeepSpeed optimizer
model_engine, optimizer, _, _ = deepspeed.initialize(
                                    args=args, model=model,
                                    model_parameters=parameters,
                                    optimizer=optimizer)
```

```
for step, batch in enumerate(data_loader):
    #forward() method
    loss = model_engine(batch)

    #runs backpropagation
    model_engine.backward(loss)

    #weight update
    model_engine.step()
```

```
{
   "zero_optimization": {
      "stage":2,
      "contiguous_gradients": true,
      "overlap_comm": true,
      "reduce_scatter": true,
      "reduce_bucket_size": 5e8,
      "allgather_bucket_size": 5e8
   }
}
```

```
# SLURM Job submission
srun train.py -b 28 -s 200 --image-size 288
 —deepspeed --deepspeed_config
ds_config_zero2.json
```

ZeRO — Optimisation du data parallelism



	Memory Consumption		Comm Volume
	Formulation	Specific Example K=12 Ψ=7.5B N_d=64	
Baseline	$(2+2+K)*\Psi$	120GB	1x
Stage 1 P_{os}	$2\Psi+2\Psi+\frac{K*\Psi}{N_d}$	31.4GB	1x
Stage 2 P_{os+g}	$2\Psi+\frac{(2+K)*\Psi}{N_d}$	16.6GB	1x
Stage 3 P_{os+g+p}	$\frac{(2+2+K)*\Psi}{N_d}$	1.9GB	1.5x

[6]

ZeRO — Rappel sur le Distributed Data Parallelism

Communications MPI

- MPI_Alltoall
- MPI_Allgather
- MPI_Allreduce (naïf)
- MPI_Allreduce (ReduceScatter + Allgather)

ZeRO — Zero Redundancy Optimizer

Communications MPI appliqués au Data Parallelism

- Distributed Data Parallelism

ZeRO

- ZeRO-DP : Stage 1
- ZeRO-DP : Stage 2

Fused optimizers

Implémentations présentent dans *APEX*

Fusionne des ***kernels* GPU** pour économiser les opérations de lecture / écriture de mémoire

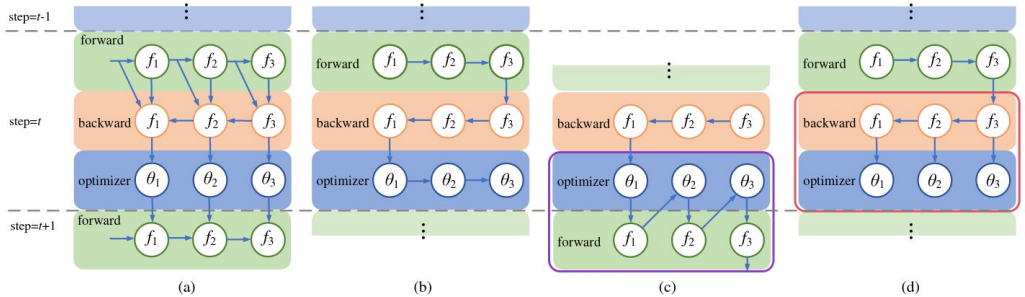
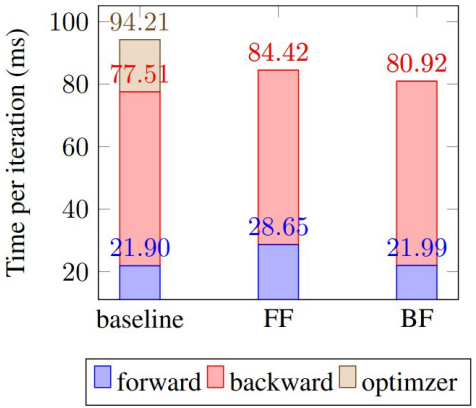


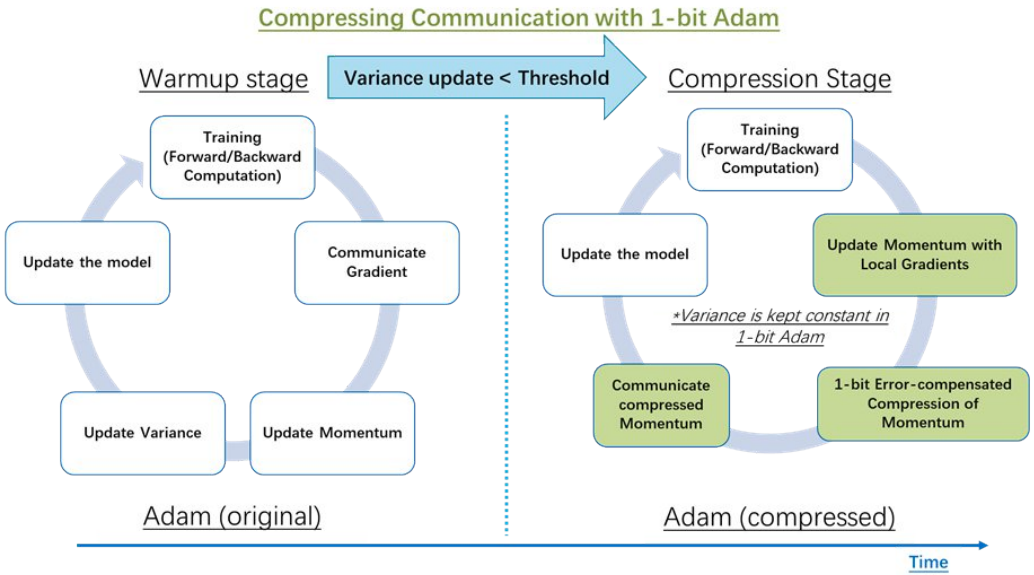
Figure 1: (a) Data dependency graph. (b) Baseline method. (c) `Forward-fusion`. (d) `Backward-fusion`. θ_i represents the trainable parameters in the layer f_i.

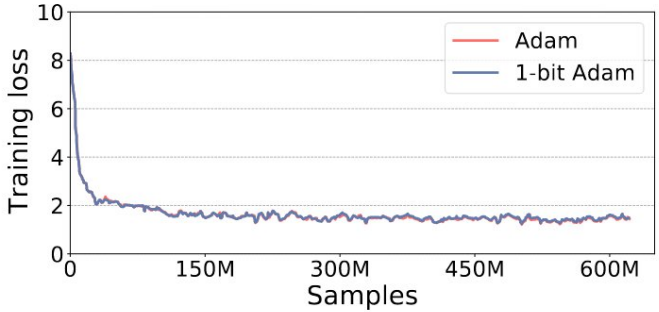


But : accélère l'étape des optimiseurs sur GPU.

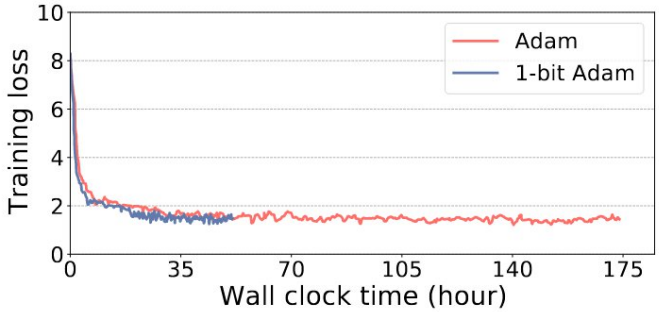
[7]

1-bit optimizers





(a) Sample-wise



(b) Time-wise

But : diminuent les communications nécessaires et donc accélère l'étape des optimiseurs pour un modèle distribué.

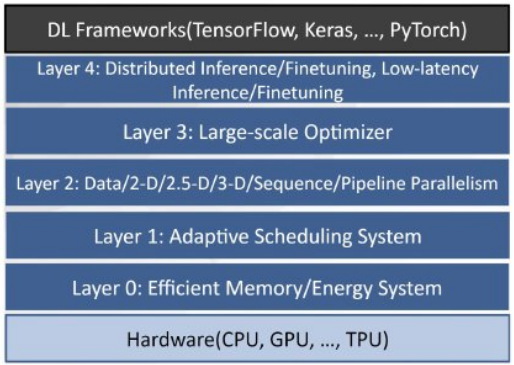
[8, 9]

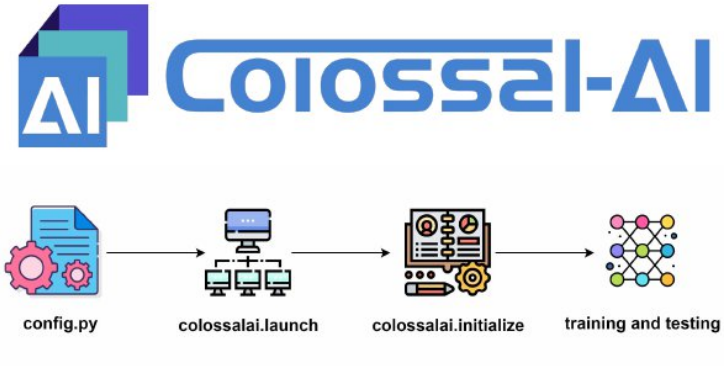
Toutes les applications de Deepspeed

- **Distributed Training with Mixed Precision**
  - 16-bit mixed precision
  - Single-GPU/Multi-GPU/Multi-Node
- **Model Parallelism**
  - Support for Custom Model Parallelism
  - **Integration with Megatron-LM**
- **Pipeline Parallelism**
  - 3D Parallelism
- **The Zero Redundancy Optimizer (ZeRO)**
  - Optimizer State and Gradient Partitioning
  - Activation Partitioning
  - Constant Buffer Optimization
  - Contiguous Memory Optimization
- ZeRO-Offload
  - Leverage both CPU/GPU memory for model training
  - Support 10B model training on a single GPU
- Ultra-fast dense transformer kernels
- Sparse attention
  - Memory- and compute-efficient sparse kernels
  - Support 10x longer sequences than dense
  - Flexible support to different sparse structures
- **1-bit Adam and 1-bit LAMB**
  - Custom communication collective
  - Up to 5x communication volume saving

- Additional Memory and Bandwidth Optimizations
  - Smart Gradient Accumulation
  - Communication/Computation Overlap
- Training Features
  - Simplified training API
  - Gradient Clipping
  - Automatic loss scaling with mixed precision
- Training Optimizers
  - Fused Adam optimizer and arbitrary torch.optim.Optimizer
  - Memory bandwidth optimized FP16 Optimizer
  - Large Batch Training with LAMB Optimizer
  - Memory efficient Training with ZeRO Optimizer
  - CPU-Adam
- Training Agnostic Checkpointing
- Advanced Parameter Search
  - Learning Rate Range Test
  - 1Cycle Learning Rate Schedule
- Simplified Data Loader
- Performance Analysis and Debugging

Colossal-AI





- Tensor Parallelism 1D, 2D, 2.5D, 3D

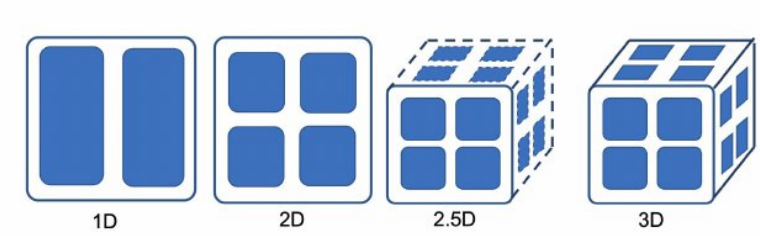
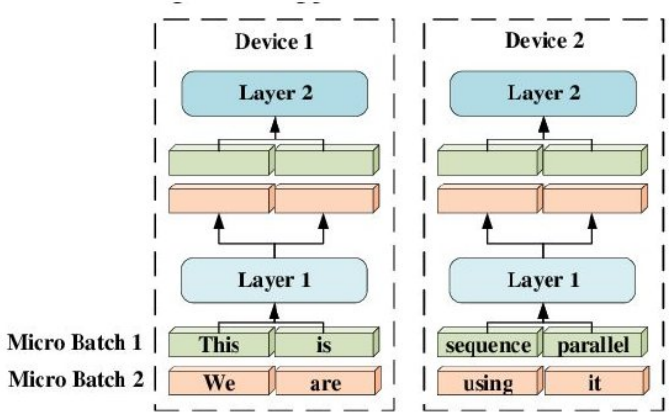


Figure 1: Tensor parallellism including 1D, 2D, 2.5D and 3D tensor splitting

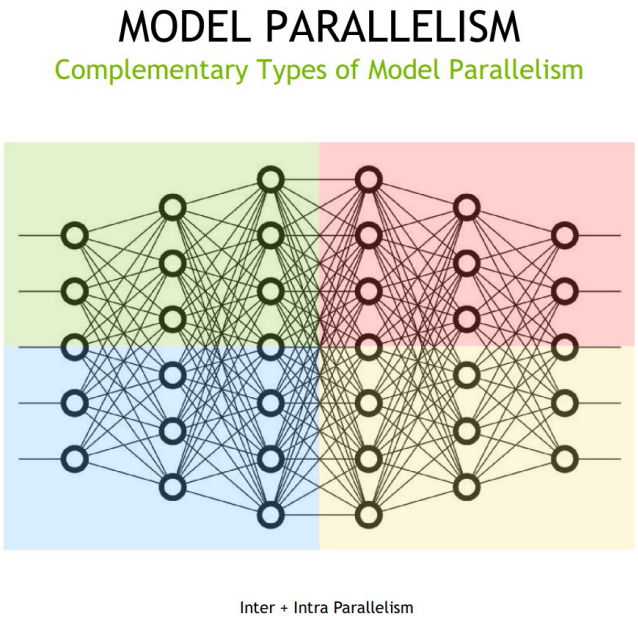
- Sequence Parallelism



[16, 17]

Megatron-LM

Model Parallelism de GPU NVIDIA (tensor and pipeline) efficace en multi-nœud pour le *pre-training* de *Transformer* comme GPT, BERT, et T5 utilisant la *mixed precision*.



Model size	Hidden size	Number of layers	Number of parameters (billion)	Model-parallel size	Number of GPUs	Batch size	Achieved teraFlOPs per GPU	Percentage of theoretical peak FLOPs	Achieved aggregate petaFLOPs
1.7B	2304	24	1.7	1	32	512	137	44%	4.4
3.6B	3072	30	3.6	2	64	512	138	44%	8.8
7.5B	4096	36	7.5	4	128	512	142	46%	18.2
18B	6144	40	18.4	8	256	1024	135	43%	34.6
39B	8192	48	39.1	16	512	1536	138	44%	70.8
76B	10240	60	76.1	32	1024	1792	140	45%	143.8
145B	12288	80	145.6	64	1536	2304	148	47%	227.1
310B	16384	96	310.1	128	1920	2160	155	50%	297.4
530B	20480	105	529.6	280	2520	2520	163	52%	410.2
1T	25600	128	1008.0	512	3072	3072	163	52%	502.0

La colonne *Model-parallel size* décrit un degré de *Tensor Parallelism* et de *Pipeline Parallelism* combinés

Pour les nombres supérieurs à 8, un *Tensor Parallelism* de taille 8 est typiquement utilisé. Ainsi, par exemple, le modèle de *145B* indique une taille de *Model Parallelism* totale de 64, ce qui signifie que cette configuration a utilisé TP=8 et PP=8.

Vision Transformers

Transformers ◂

Vision Transformers ◂

CoAtNet ◂

Vision Transformers >> Resnet-50: 25M

Rank	Model	Top 1 Accuracy	Top 5 Accuracy	Number of params	Extra Training Data	Paper	Code	Result	Year	Tags
1	**CoAtNet-7**	90.88%		2440M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer, JFT-3B
2	**ViT-G/14**	90.45%		1843M	✓	Scaling Vision Transformers			2021	Transformer, JFT-3B
3	**CoAtNet-6**	90.45%		1470M	✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer, JFT-3B
4	**V-MoE-15B** (Every-2)	90.35%		14700M	✓	Scaling Vision with Sparse Mixture of Experts			2021	Transformer
5	**SwinV2-G**	90.17%			✓	Swin Transformer V2: Scaling Up Capacity and Resolution			2021	Transformer
6	**Florence-CoSwin-H**	90.05%	99.02%		✓	Florence: A New Foundation Model for Computer Vision			2021	Transformer
7	**TokenLearner L/8** (24+11)	88.87%		460M	✓	TokenLearner: What Can 8 Learned Tokens Do for Images and Videos?			2021	Transformer, JFT-300M
8	**MViT-H, 512^2** (IN22K-pretrain)	88.8%		667M	✓	Improved Multiscale Vision Transformers for Classification and Detection			2021	Transformer, ImageNet-22k, MViT

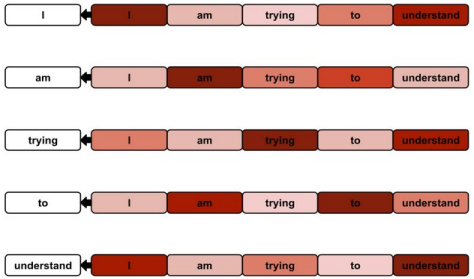
[Paperwithcode]

Le premier Transformer

Attention Is All You Need

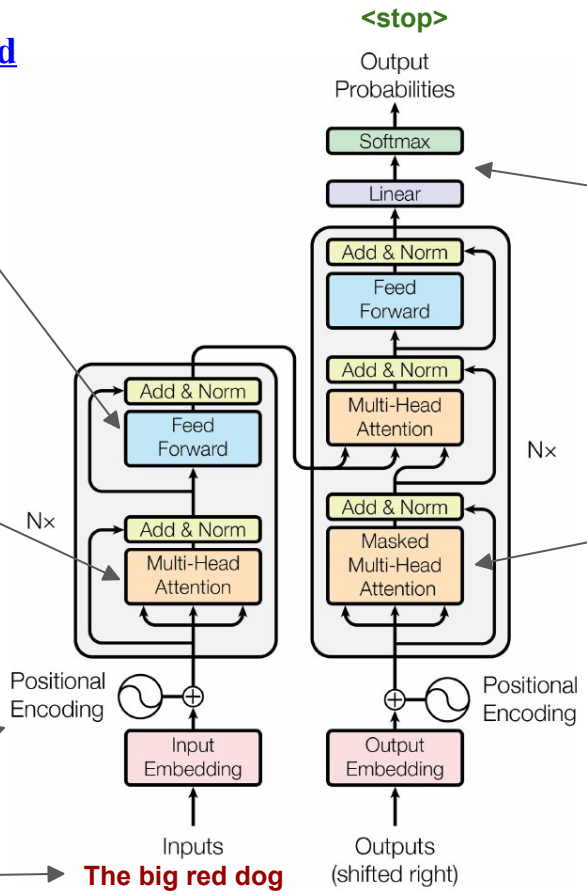
Simple couche de réseaux de neurones

Mécanisme d'attention (encoder)



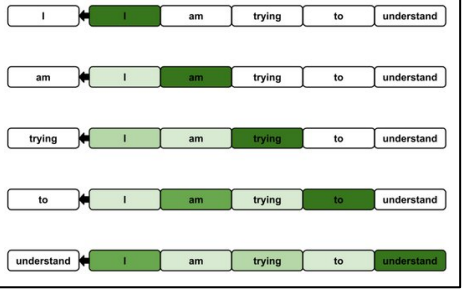
Ajout des vecteurs positions à la séquences

Séquence de vecteurs (word embeddings)



Simple classification sur tout le vocabulaire

Mécanisme d'attention (decoder)

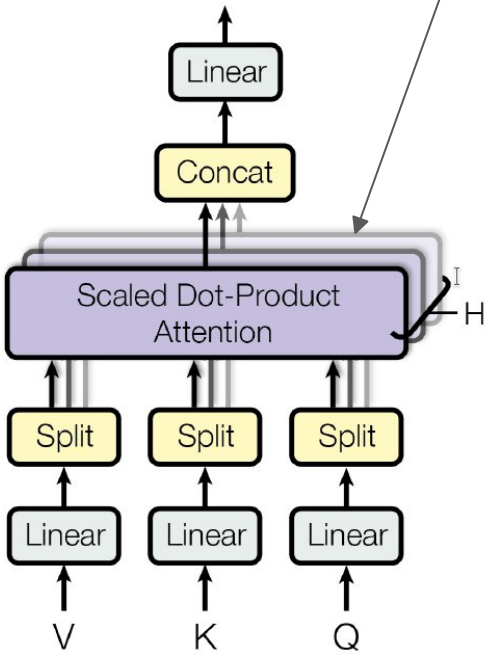


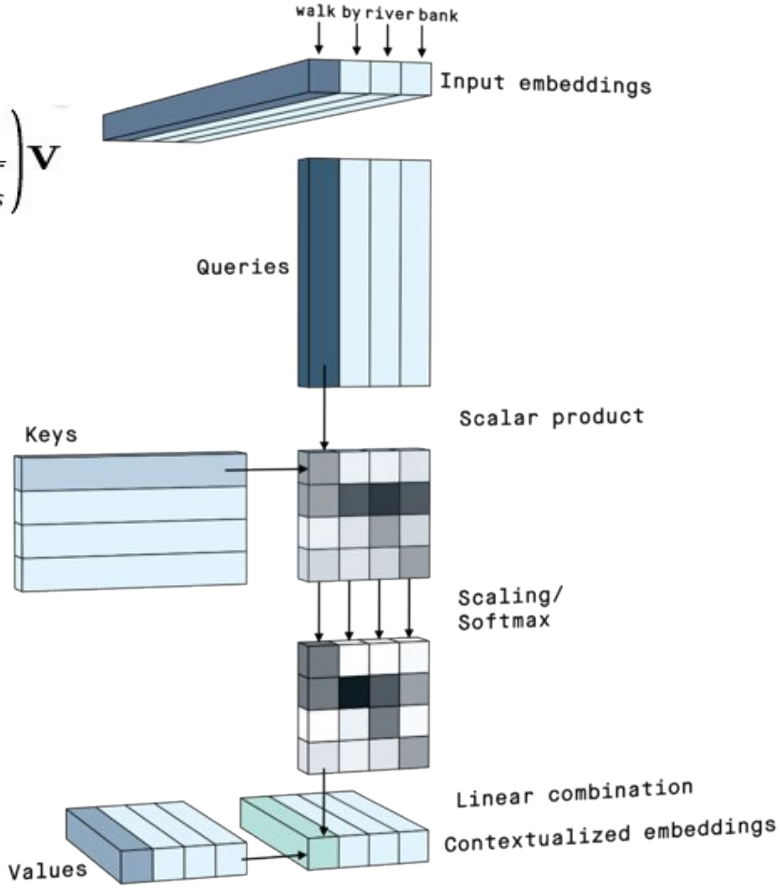
[10]

Le mécanisme de Self-Attention

Scaled Dot-Product Attention

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{keys}}}\right)\mathbf{V}$$



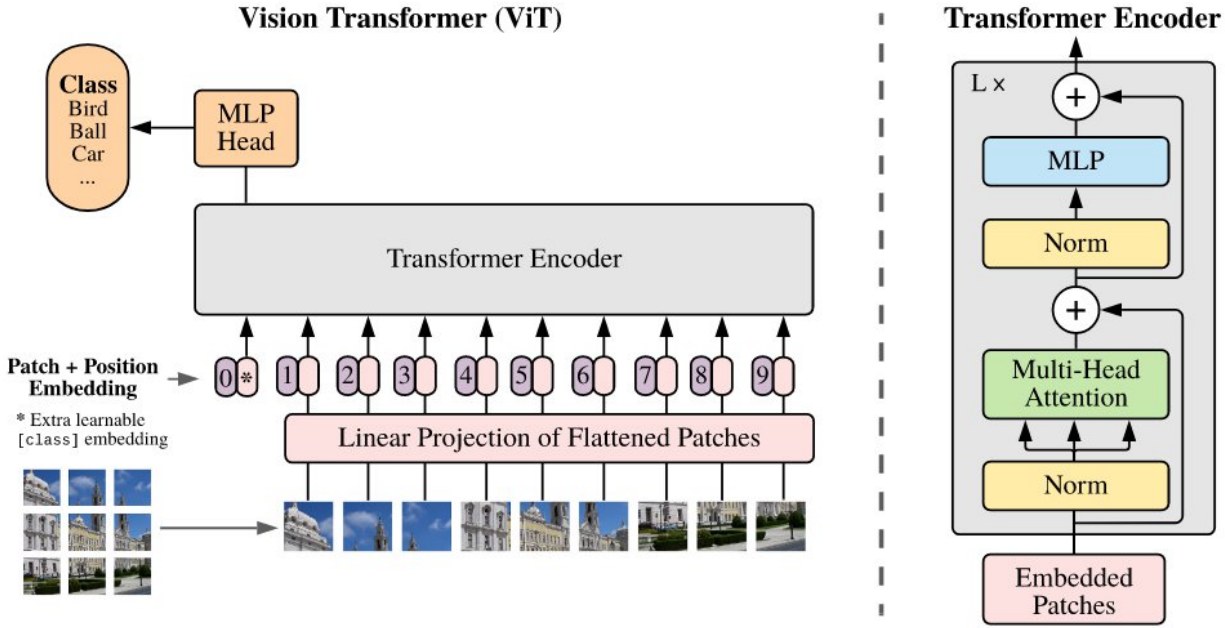


[10, 11]

Les Transformers

- Transforment la séquence entière (contrairement aux CNN et aux RNN)
- Possèdent un nombre conséquent de poids
- Nécessitent de gros *datasets*

Vision Transformer (ViT)



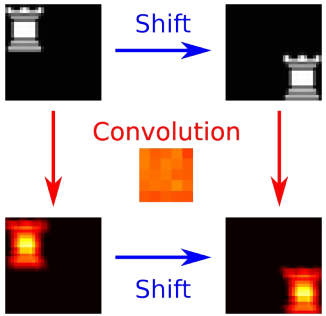
- Images découpées en *patch*
- *Patchs* séquencés avec un *Position embedding*
- Ajout d'un "classification token" pour réaliser la classification finale

[12]

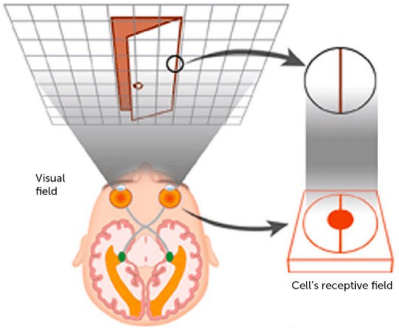
CoAtNet

“Marrying Convolution and Attention for All Data Sizes”

ConvNet
Translation Equivariance



Self-Attention Net
Global Receptive Field
Context comprehension



No patching !!

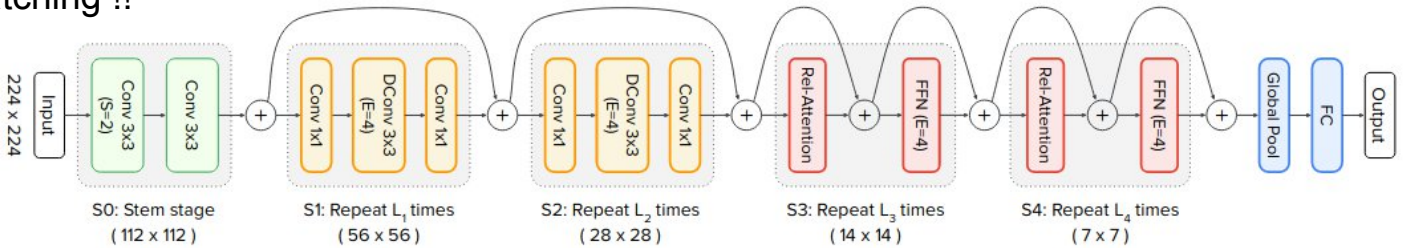


Figure 4: Overview of the proposed CoAtNet.

[13, 14, 15]

CoAtNet - Résultats

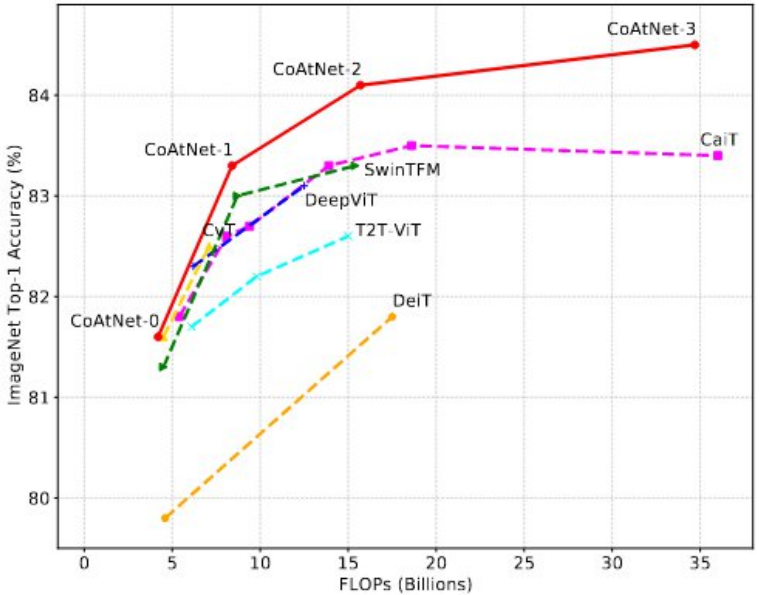


Figure 2: Accuracy-to-FLOPs scaling curve under ImageNet-1K only setting at 224x224.

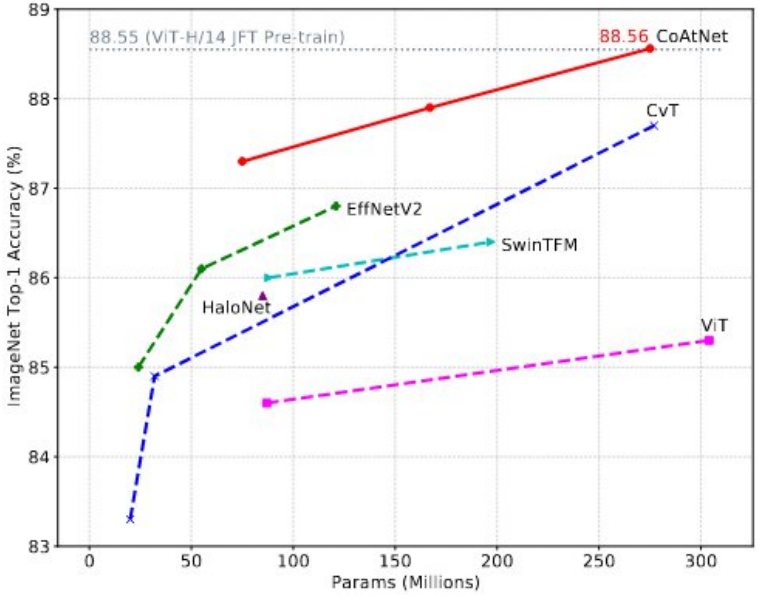
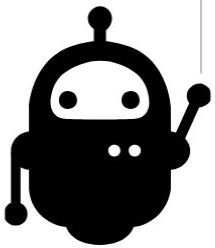


Figure 3: Accuracy-to-Params scaling curve under ImageNet-21K ⇒ ImageNet-1K setting.

[13]

TP : CoAtNet-7

```
local:~$ ssh jean-zay

jz:~$ module load pytorch-gpu/py3/1.11.0
jz:~$ idrjup $WORK
```



- Limite du *Data Parallelism* avec CoAtNet
- Implémenter ZeRO
- Implementer le Pipeline Parallelism
- Recherche du meilleur compromis

Références des images utilisées

1. HuggingFace 2021, https://huggingface.co/blog/large-language-models
2. Nicolae, Bogdan, et al. "Deepfreeze: Towards scalable asynchronous checkpointing of deep learning models." *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020.
3. FairScale authors. (2021). FairScale: A general purpose modular PyTorch library for high performance and large scale training. https://fairscale.readthedocs.io/en/latest/deep_dive/pipeline_parallelism.html
4. Fan, Shiqing, et al. "DAPPLE: A pipelined data parallel approach for training large models." *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 2021.
5. Narayanan, Deepak, et al. "PipeDream: Generalized pipeline parallelism for DNN training." *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 2019.
6. Rajbhandari, Samyam, et al. "Zero: Memory optimizations toward training trillion parameter models." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.
7. Jiang, Zixuan, et al. "Optimizer Fusion: Efficient Training with Better Locality and Parallelism." *arXiv preprint arXiv:2104.00237* (2021).
8. Deepspeed 2020, https://www.deepspeed.ai/2020/09/08/onebit-adam-blog-post.html
9. Tang, Hanlin, et al. "1-bit adam: Communication efficient large-scale training with adam's convergence speed." *International Conference on Machine Learning*. PMLR, 2021.
10. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
11. Peltarion, https://peltarion.com/blog/data-science/self-attention-video
12. Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).
13. Dai, Zihang, et al. "Coatnet: Marrying convolution and attention for all data sizes." *Advances in Neural Information Processing Systems* 34 (2021): 3965-3977.
14. Medium, https://medium.com/@oskyhn_77789/current-convolutional-neural-networks-are-not-translation-equivariant-2f04bb9062e3
15. AI Summer, https://theaisummer.com/receptive-field/
16. Bian, Zhengda, et al. "Colossal-AI: A unified deep learning system for large-scale parallel training." *arXiv preprint arXiv:2110.14883* (2021).
17. https://medium.com/@hpcaitech/colossal-ai-a-unified-deep-learning-system-for-large-scale-parallel-training-2fed5df097c0