Deep Learning Optimisé - Jean Zay

Résultats Imagenet Race et bonnes pratiques

INSTITUT DU DÉVELOPPEMENT ET DES RESSOURCES EN INFORMATIQUE SCIENTIFIQUE

DLO-JZ

5ème partie de la formation de l'IDRIS.

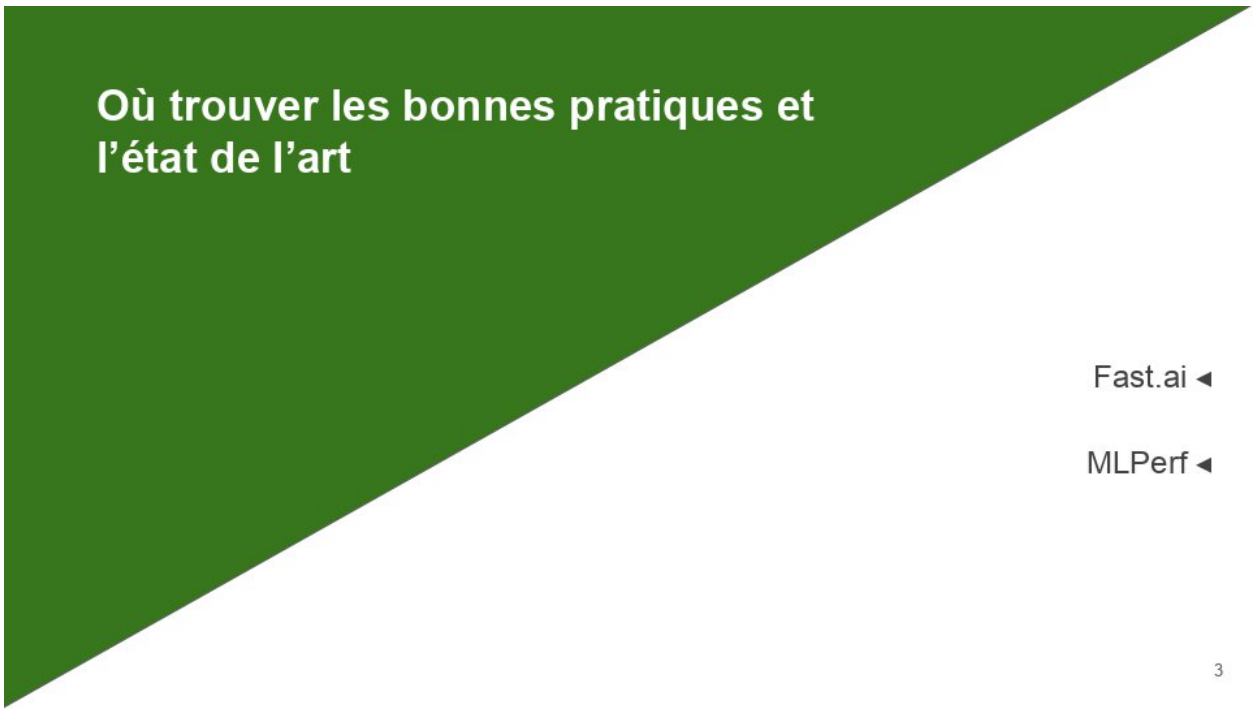
Diapositives commentées.

Auteur : Bertrand Cabot

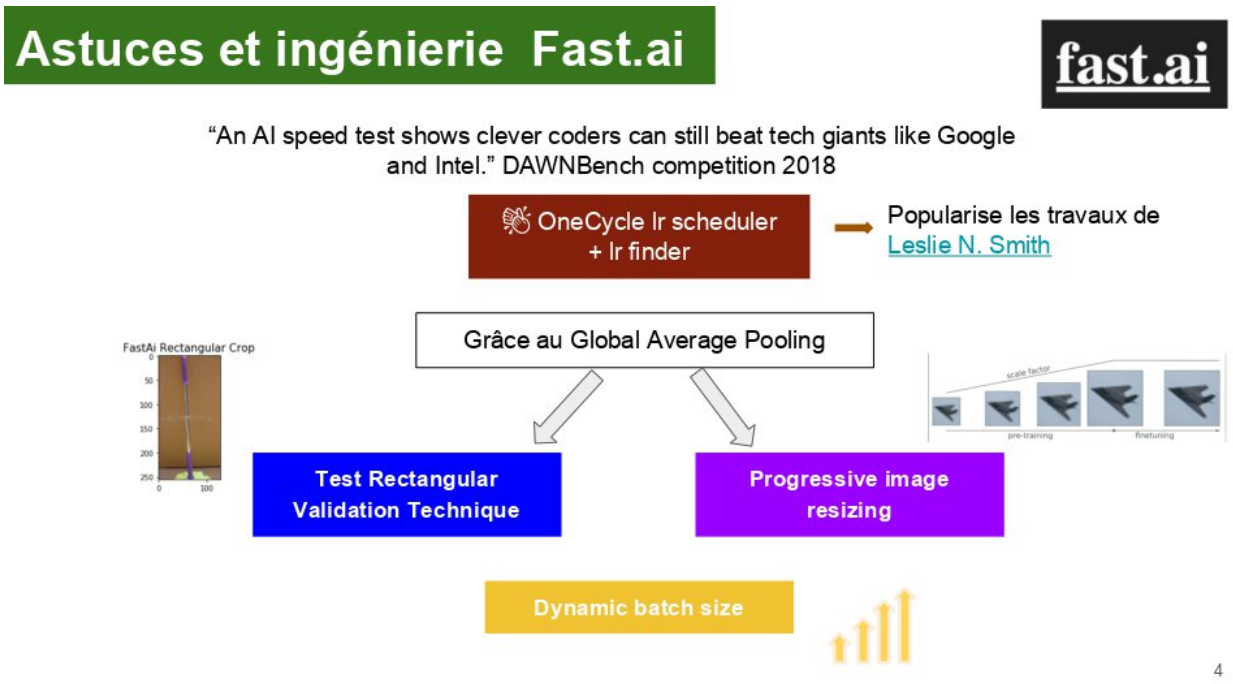
Juin 2023

Chapitres :

- Où trouver les bonnes pratiques et l'état de l'art ?
- Frameworks et Optimisations du compileur



Cette partie présente quelques sources d'informations importantes sur les bonnes pratiques et l'état de l'art pour l'accélération de l'apprentissage en Deep Learning.



Fast.ai est une initiative de personnalités du *Queensland University of Technology* ayant pour objectif de démocratiser le Deep Learning grâce à une communauté de développeurs, des cours en ligne, une sur-couche méthodologique de PyTorch avec toutes les bonnes pratiques de l'apprentissage en Deep Learning.
Notamment en 2018, la communauté a participé et remporté la compétition *DAWNBench* devant les géants de la *tech*, montrant ainsi qu'avec des bonnes pratiques, de la méthodologie, de l'intuition et des astuces, il était possible de surclasser les moyens énormes *hardware* des géants de la *tech*.
Ils ont publiés ensuite la recette pour entraîner *« Imagenet from scratch* pour seulement 40 dollars » ou en 18 minutes sur 256 GPU (équivalence du prix de la location d'une instance sur AWS).

Pour ce qui nous intéresse par rapport à notre présentation, ils ont popularisé les travaux de *Leslie N. Smitch* (vus ce matin dans DLO-JZ4) notamment les *OneCycle lr scheduler* et le *lr finder*.
Ils ont poussé le fait d'utiliser le *Global Average Pooling* à la fin des architectures CNN pour pouvoir jouer sur le format des images pendant l'apprentissage. Ainsi, 2 astuces ont été mises en avant :

- l'utilisation d'image rectangulaire pour *booster* la validation
- La possibilité d'agrandir petit à petit la taille des images lors de l'apprentissage pour accélérer au début et affiner à la fin.

De plus, ils ont popularisé aussi le *Dynamic batch size* qui consiste à augmenter la taille du batch progressivement pendant l'apprentissage. Cela peut remplacer ou compléter le *lr scheduler* que nous avons vu.

Tout cela, vers 2018, a été un apport conséquent pour l'accélération et la précision de l'apprentissage en Deep Learning.

ML Perf - Référence pour le Supercomputing en IA

Référence pour l'industrie

- Hardware
- Framework de Deep Learning
- Techniques SOTA

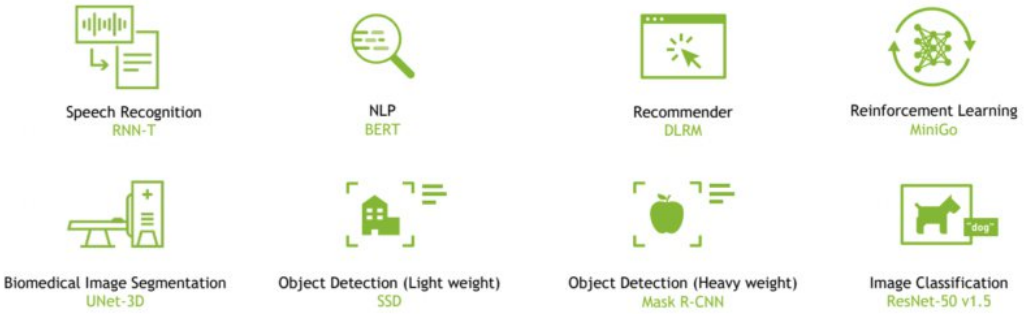
5

Cependant pour avoir une vue plus globale et actuelle sur l'optimisation de l'accélération des différentes applications en Deep Learning, le suite *ML Perf* devient une référence incontournable.

ML Perf est un ensemble de *benchmark* de Deep Learning permettant d'abord un affichage publicitaire des constructeurs de matériel pour comparer les performances des *Hardware*, des *Software* pour l'ensemble des applications du *Deep Learning*. Ce qui va nous permettre, on va le voir ensuite, de rassembler toutes les bonnes pratiques et les méthodologies pour accélérer un apprentissage.

ML Perf - Benchmarks

Training

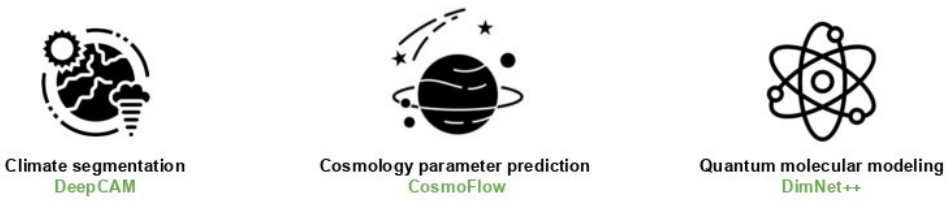


6

La suite de *benchmark ML Perf* propose 8 applications classiques d'apprentissage de Deep Learning.

ML Perf - Benchmarks

Training HPC



Inference :

- Datacenter
- Edge
- Mobile
- Tiny

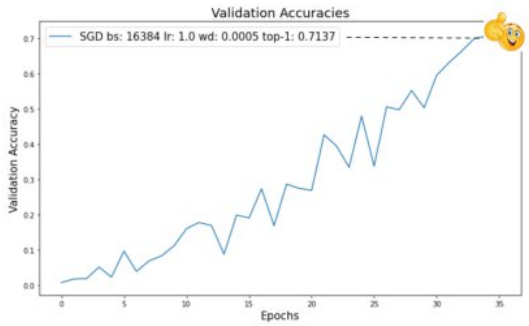
7

Elle comprend aussi :

- 3 applications d'apprentissage de Deep Learning orientées "calcul scientifique",
- Un nombre d'applications d'inférence selon la taille des équipements.

ML Perf - Benchmarks

Training rule



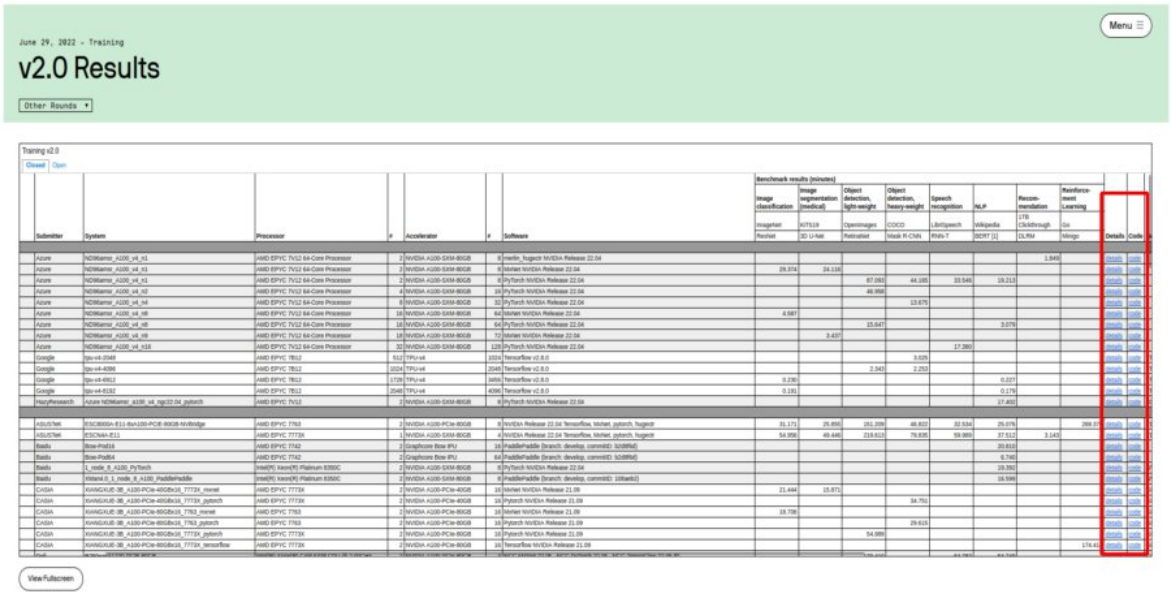
A metric threshold to reach

Time Results

8

Pour valider l'apprentissage et pouvoir publier un temps d'apprentissage dans le tableau de résultat, il faut atteindre un seuil prédéfini pour la métrique d'évaluation de l'apprentissage.

ML Perf – sources pour connaître l'état de l'art



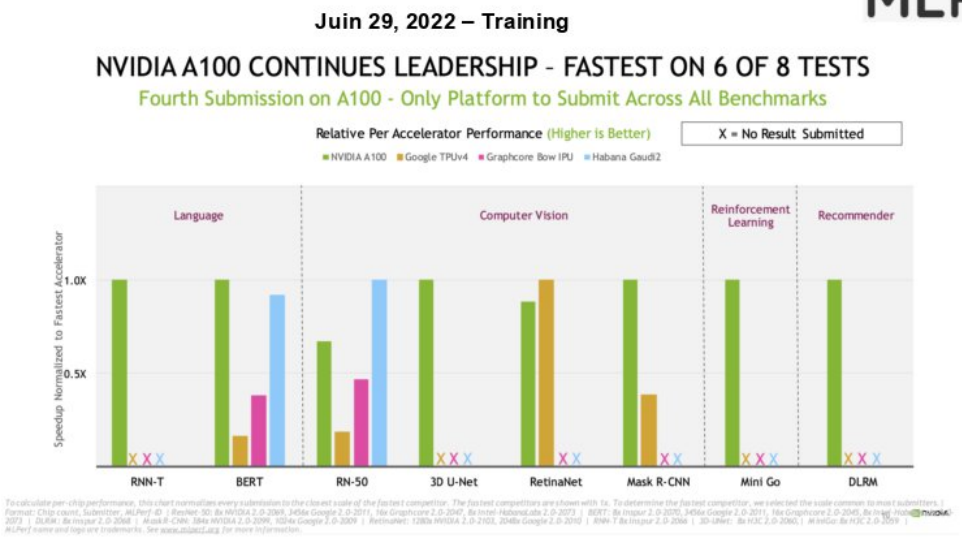
Cela met effectivement en avant les performances des équipements et des frameworks, et pousse surtout la recherche des géants de la *tech*.

La plupart des résultats sont donnés dans la partie *closed*, où un certain nombre de paramètres sont fixés (le framework, l'*optimizer*, etc), ce qui permet surtout de comparer les systèmes entre eux.

Mais il existe aussi une section *open* qui permet à des entités de valoriser leur solution plutôt software ou algorithmique.

Mais pour nous, simple utilisateur, cela nous permet d'avoir une source de codes et de documentations importante sur les bonnes pratiques et méthodologies de l'accélération distribuée, très actualisée.

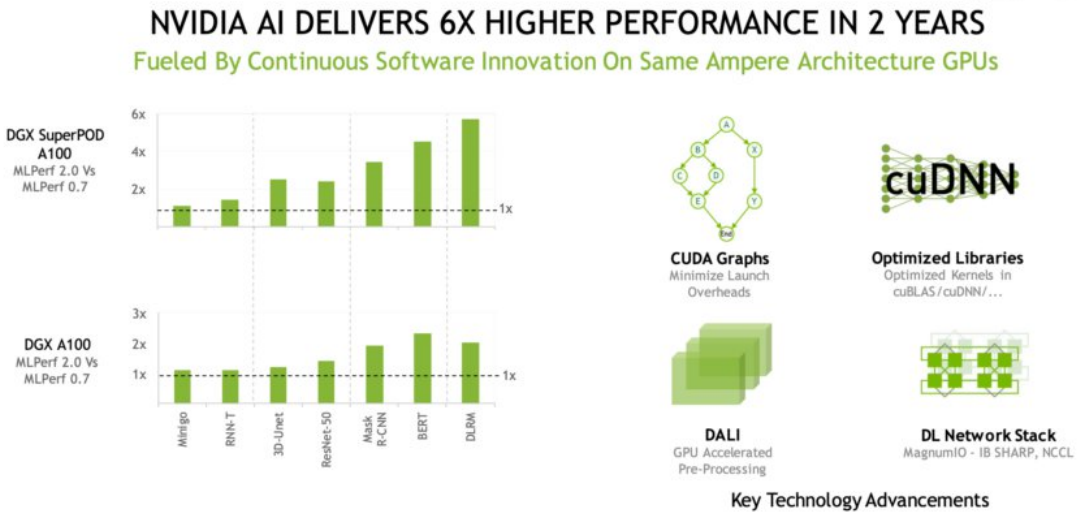
ML Perf - Result v2.0



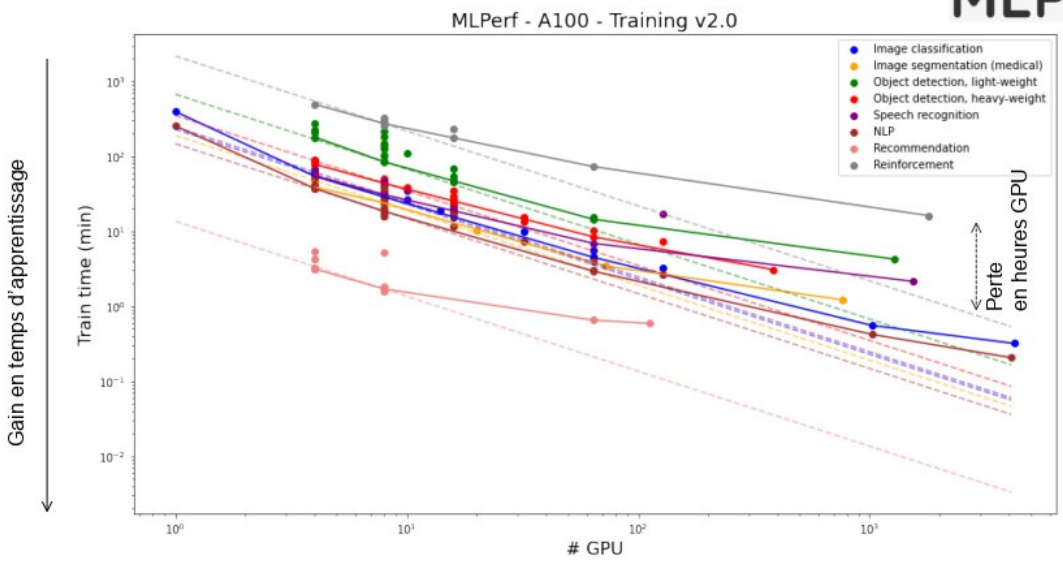
Les résultats permettent de comparer les différents types d'accélérateur IA.

Les résultats permettent aussi de suivre l'évolution du matériel d'accélération (GPU Nvidia) en *Deep Learning*.

ML Perf - Évolution



ML Perf - Scaling



En récupérant les résultats MLPerf, on peut dessiner une représentation du *scaling* de la distribution et des bonnes pratiques associées, sur les supers calculateurs, en fonction du type d'application.

On peut ainsi observer que certaines applications sont sujettes à une distribution plus large sur un nombre important d'accélérateurs GPU par rapport aux autres applications moins sujettes (Classification d'image, NLP).

Dans tous les cas, utiliser plus de GPU en *Data Parallelism* réduit le temps global d'apprentissage. Cependant l'écart entre la courbe réelle et la courbe idéale montre une consommation d'heure GPU et donc une consommation électrique plus importante à résultat équivalent.

De ce point de vue économie énergétique, on voit que pour les Benchmarks MLPerf 1 ou 2 nœuds de calcul correspond au meilleur compromis pour la consommation énergétique.

Torch IMage Models

doc : https://timm.fast.ai/

TIMM is a library containing SOTA **computer vision models**, layers, utilities, optimizers, schedulers, data-loaders, augmentations, and training/evaluation scripts.

papers with code :
https://paperswithcode.com/lib/timm

Documentation

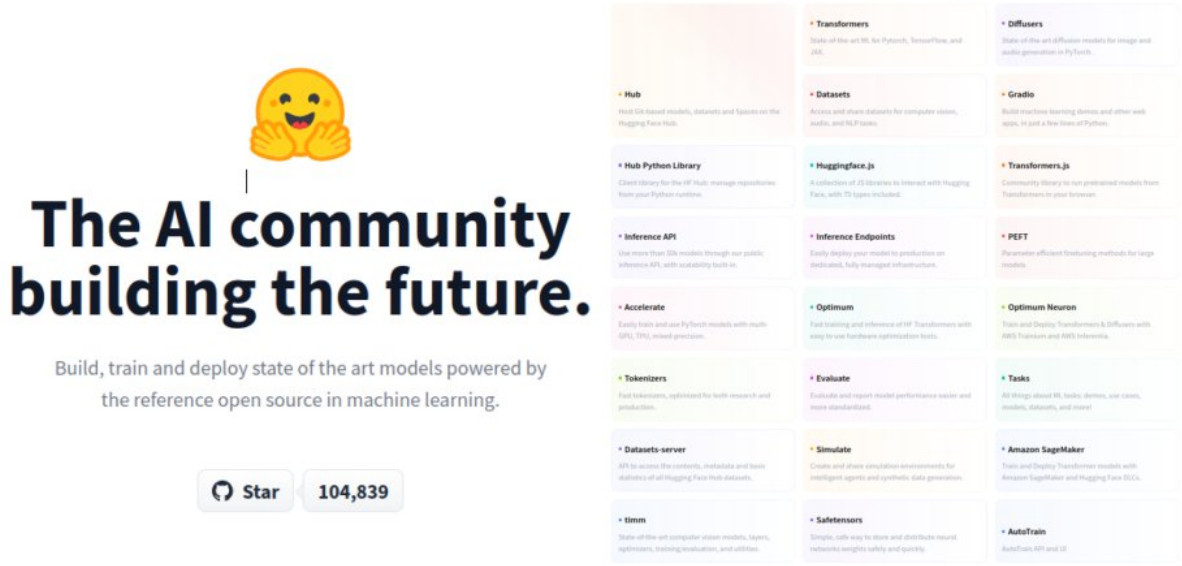
https://huggingface.co/docs/timm/index

12

La librairie TIMM (Torch Image Models) initiative de Ross Wightman implémente et met à disposition la plupart des modèles “état de l’art” en *computer vision,* des *optimizers*, des *schedulers*, des techniques de d*ata augmentation*, ...

TIMM est une référence en *computer vision*. Récemment, la librairie a été intégrée à la bibliothèque *Hugging Face*.

Hugging Face

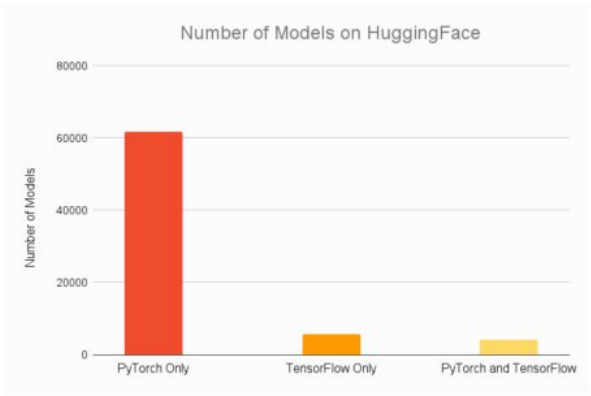


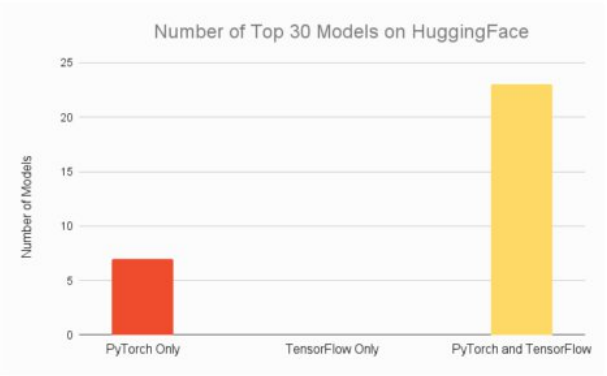
Hugging Face est devenu depuis l'avènement des Transformer la bibliothèque, le zoo de modèle, le Hub de référence pour la recherche ouverte en *Deep Learning*.

Les librairies applicatives notables parmi un ensemble de plus en plus étoffé sont : « *Transformers* » qui permet l'accès facile à la plupart des modèles disponible dans le zoo Hugging Face, et *« Accelerate* » qui permet d'accéder facilement à toutes les techniques de *scaling* abordées durant cette formation.

Cette partie discute des différents *Frameworks* de *Deep learning* et de l'optimisation des différentes méthodes de compilation (JIT, AOT, ...)

Pytorch vs Tensorflow



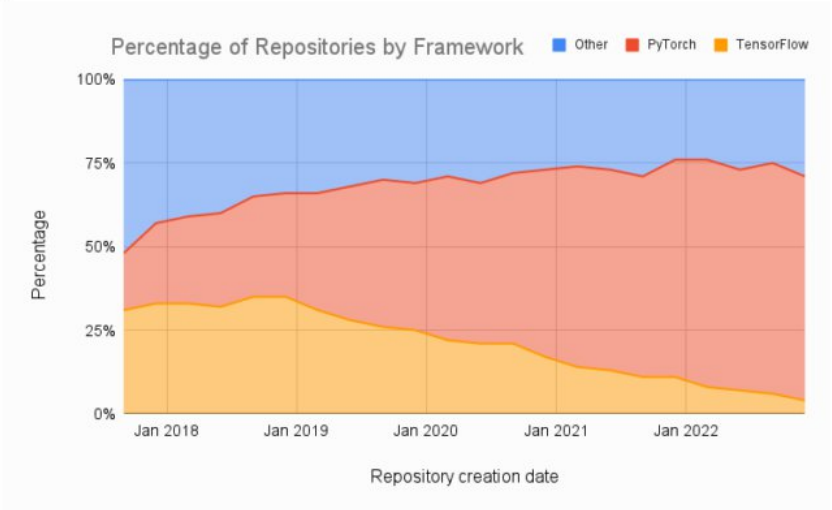


https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/ 15

Dans la bibliothèque *Hugging Face, Pytorch* est largement sur-représenté par rapport à *Tensorflow.*

Pytorch vs Tensorflow

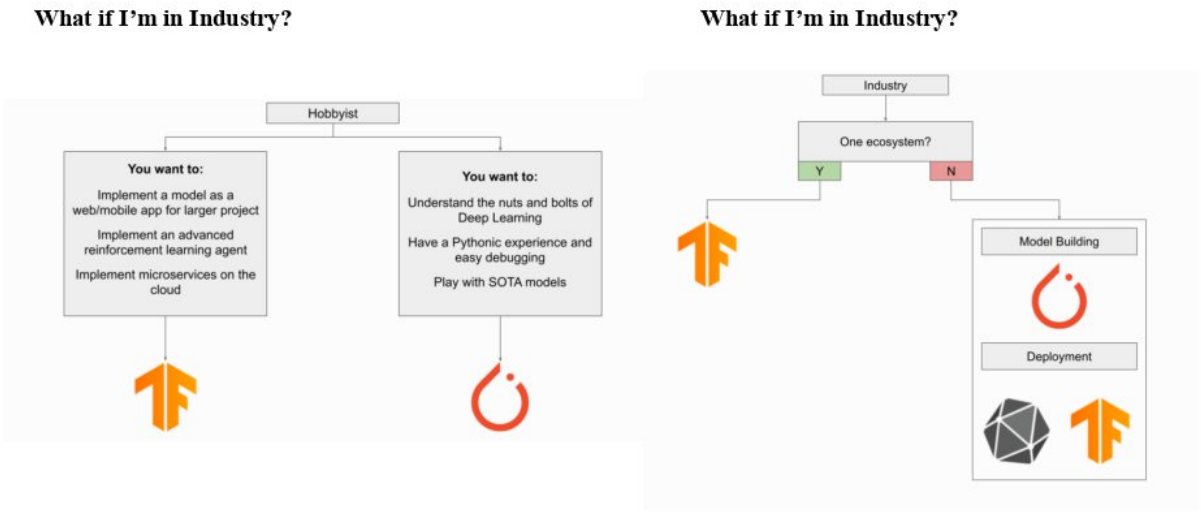
Paper With Code :



https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/ 16

Pytorch prend de plus en plus d'importance dans la recherche en IA selon les publications de dépôts sur *paperwithcode*. Pytorch prend une place de plus importante dans la recherche en IA utilisant des super calculateurs. Excepté Google (Google brain + Deepmind), la plupart des géants de la tech utilise Pytorch aujourd'hui.

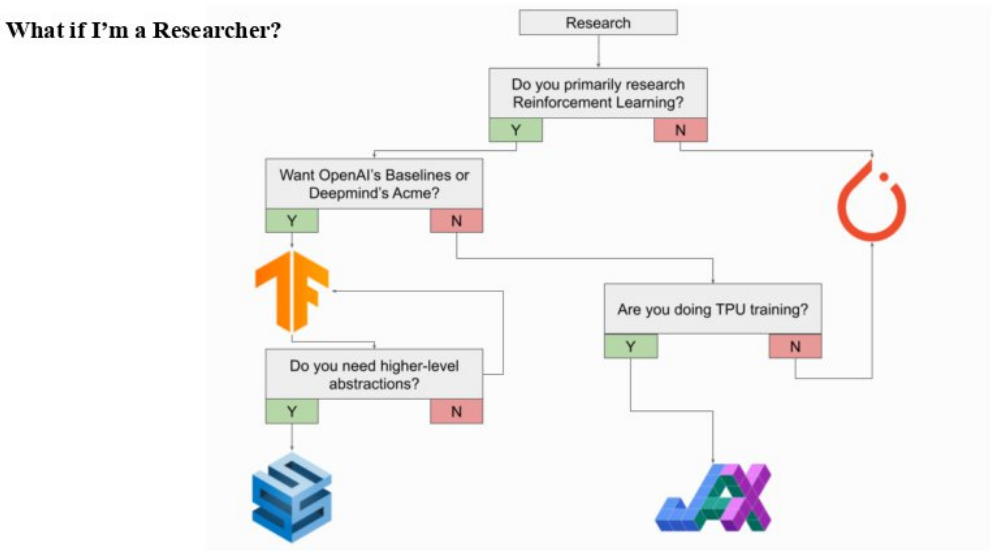
Pytorch vs Tensorflow



https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/ 17

Pour les amateurs ou les industries, le choix se fera en faveur de *Tensorflow* si ils souhaitent appliquer et déployer un réseau de neurones, et de Pytorch si ils souhaitent participer à la recherche.

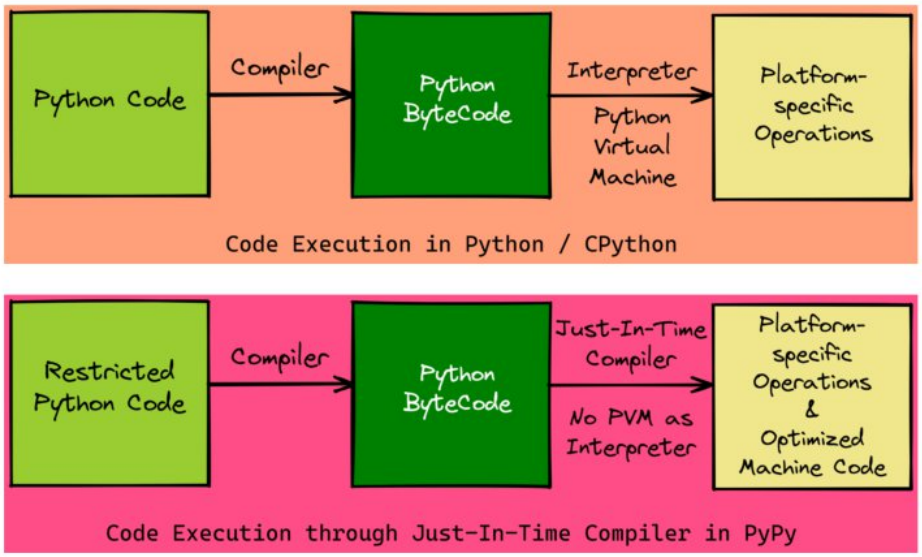
Pytorch vs Tensorflow vs JAX



https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/ 18

Pour les chercheurs, le choix se fera sur Pytorch ou JAX particulièrement si ils utilisent des TPU.

JIT : Just In Time Compilation



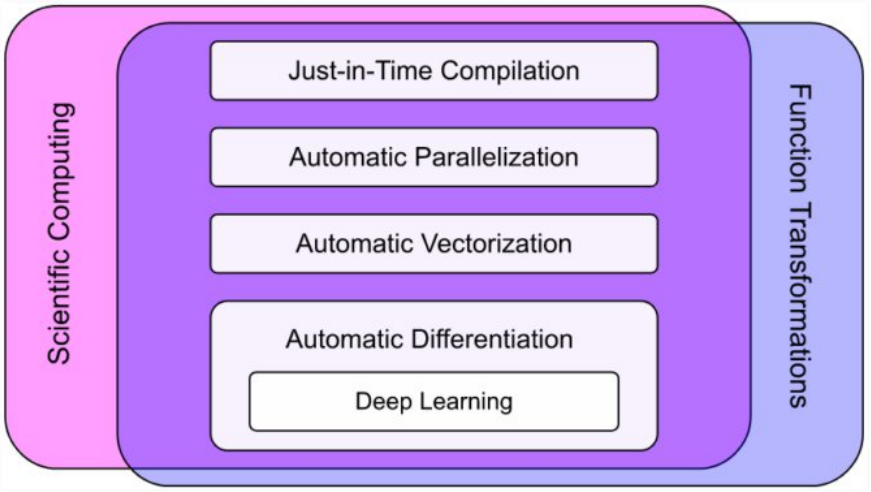
19

La compilation *Just-In-Time* est une optimisation de la compilation d'un code python.

Classiquement un code python est compilé en *ByteCode* interprétable ensuite dans sa globalité par une machine virtuelle python.

La compilation *Just-In-Time* compile ligne par ligne le ByteCode en temps réel. Cela permet une accélération. Cependant cette compilation sera limité aux lignes de ByteCode et aux plateformes (device) dédiées.

Jax



JAX lies at the intersection of Scientific Computing and Function Transformations, yielding a wide range of capability beyond the ability to train Deep Learning models

20

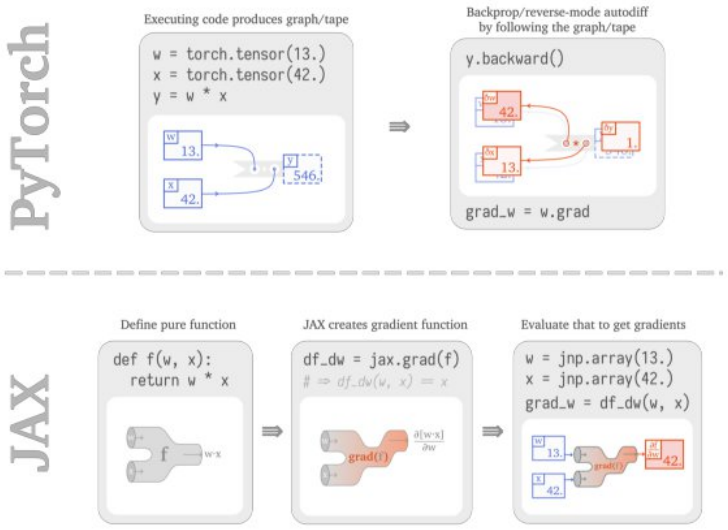
JAX est un framework pour l'apprentissage automatique qui vous permet d'utiliser Python et NumPy pour créer et entraîner des réseaux de neurones.

JAX inclut la compilation juste-à-temps (JIT), qui est une technique de compilation de code à la volée. La compilation JIT peut être utilisée pour améliorer les performances du code numérique, tel que le code utilisé dans l'apprentissage en profondeur.

JAX utilise un compilateur JIT appelé XLA, également utilisé par TensorFlow. XLA est un compilateur hautes performances capable d'optimiser le code pour une variété d'architectures, y compris les processeurs, les GPU, les TPU et les accélérateurs matériels personnalisés.

JAX dispose également d'une fonctionnalité appelée "parallélisation automatique". Cela signifie que JAX peut automatiquement paralléliser votre code sur plusieurs CPU, GPU ou TPU. Cela peut être utile pour former de très grands réseaux de neurones.

Pytorch vs Jax



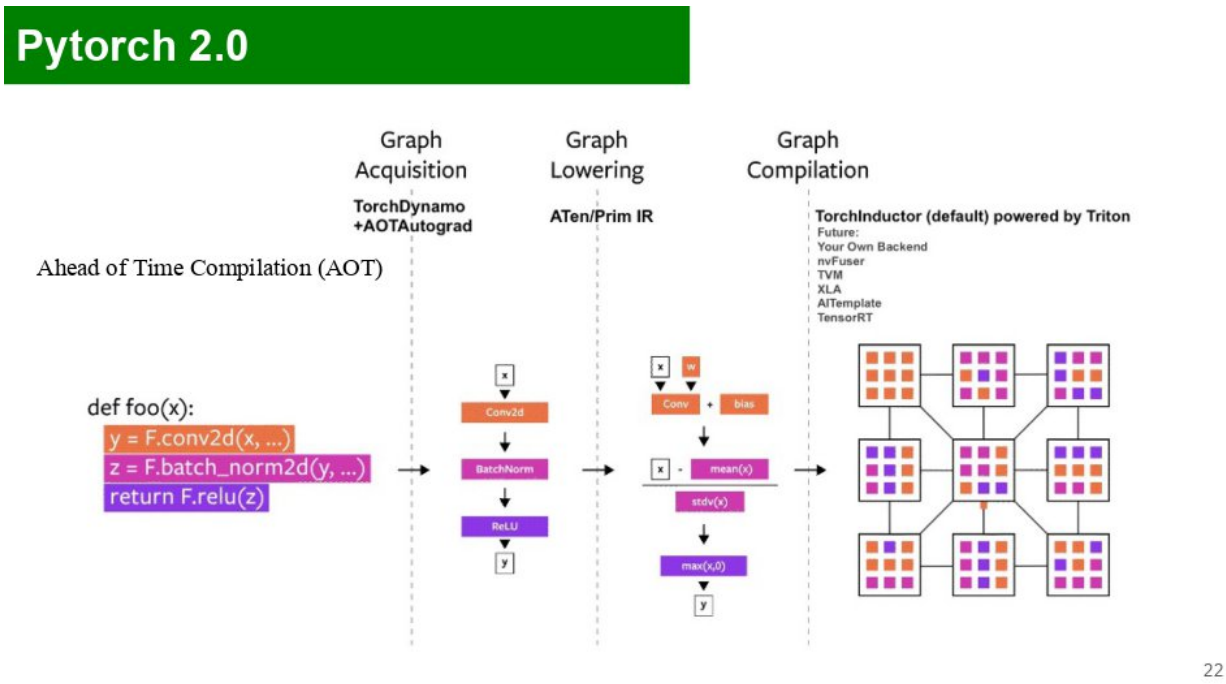
Pros :
Pythonic, dynamic, popular framework,
NVIDIA GPU oriented, Jean Zay adapted
Cons :
Slower compare to some other frameworks (Jax, Mxnet, ...)

Pros :
Fast Computation
numpy-like, functional programming, Hessian computation (2nd order) efficiency
Cons :
Google/TPU oriented, Jean Zay unadapted

21

Pytorch est basé sur le concept de “dynamic graph” ce qui le rend très facile à manipuler et très populaire. Il est particulièrement adapté aux accélérateurs GPU Nvidia et donc à Jean Zay. Cependant, il peut être considéré comme plus lent par rapport à d’autres frameworks (JAX, MXNET, ...). La large communauté de développement qui le compose permet rapidement de corriger ses faiblesses et de s’adapter à tout type d’équipement.

JAX est basé sur le concept de "transformations de fonctions". Cela signifie que vous pouvez définir une fonction, puis JAX calculera automatiquement la dérivée de cette fonction. Ainsi il est performant pour le calcul de dérivé de second ordre aussi : ce qui ouvre des possibilités de recherche intéressante. Cependant il est orienté TPU et parfois difficile à mettre en œuvre sur Jean Zay.



Au cours des 5 dernières années, Torch a construit torch.jit.trace, TorchScript, FX tracing, Lazy Tensors. Mais aucun d'entre eux ne donnait satisfaction. Certains étaient flexibles mais pas rapides, certains étaient rapides mais pas flexibles et certains n'étaient ni rapides ni flexibles. Certains avaient une mauvaise expérience utilisateur (comme se tromper silencieusement). Alors que TorchScript était prometteur, il nécessitait des modifications substantielles du code. Ce besoin de changement substantiel dans le code en a fait un non-démarreur pour de nombreux utilisateurs de PyTorch.

Pytorch 2.0 est un tournant majeur avec Torch.compile.

Torch.compile repose sur de nouvelles technologies – TorchDynamo, AOTAutograd, PrimTorch et TorchInductor.

TorchDynamo capture les programmes PyTorch en toute sécurité à l'aide de Python Frame Evaluation Hooks et est une innovation importante qui est le résultat de 5 ans de notre R&D dans la capture sécurisée de graphes

AOTAutograd surcharge le moteur autograd de PyTorch en tant qu'autodiff de traçage pour générer des traces en arrière à l'avance.

PrimTorch canonise ~2000+ opérateurs PyTorch en un ensemble fermé de ~250 opérateurs primitifs que les développeurs peuvent cibler pour

créer un backend PyTorch complet. Cela réduit considérablement la barrière de l'écriture d'une fonctionnalité ou d'un backend PyTorch.

TorchInductor est un compilateur d'apprentissage en profondeur qui génère du code rapide pour plusieurs accélérateurs et backends. Pour les GPU NVIDIA et AMD, il utilise OpenAI Triton comme élément de construction clé.

Pytorch 2.0

```
# API NOT FINAL
# default: optimizes for large models, low compile-time
#          and no extra memory usage
torch.compile(model)

# reduce-overhead: optimizes to reduce the framework overhead
#                and uses some extra memory. Helps speed up small models
torch.compile(model, mode="reduce-overhead")

# max-autotune: optimizes to produce the fastest model,
#               but takes a very long time to compile
torch.compile(model, mode="max-autotune")
```

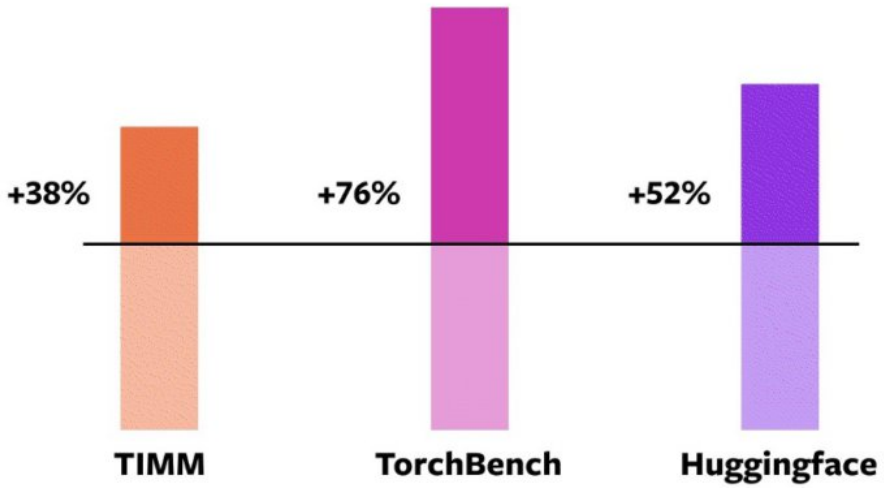
23

Le mode spécifie ce que le compilateur doit optimiser lors de la compilation.

Le mode par défaut est un préréglage qui essaie de compiler efficacement sans prendre trop de temps à compiler ou en utilisant de la mémoire supplémentaire.

D'autres modes tels que la réduction de surcharge réduisent beaucoup plus la surcharge du framework, mais coûtent une petite quantité de mémoire supplémentaire. max-autotune compile pendant longtemps, essayant de vous donner le code le plus rapide qu'il puisse générer.

Pytorch 2.0



24

Pour valider ces technologies, nous avons utilisé un ensemble diversifié de 163 modèles open source dans divers domaines d'apprentissage automatique. Nous avons soigneusement construit ce benchmark pour inclure des tâches telles que la classification d'images, la détection d'objets, la génération d'images, diverses tâches NLP telles que la modélisation du langage, les questions et réponses, la classification des séquences, les systèmes de recommandation et l'apprentissage par renforcement. Nous séparons les benchmarks en trois catégories :

- 46 modèles de HuggingFace Transformers
- 61 modèles de TIMM : une collection de modèles d'image PyTorch à la pointe de la technologie par Ross Wightman
- 56 modèles de TorchBench : un ensemble organisé de bases de code populaires provenant de github