L’entraînement de réseaux neuronaux, en pratique

Par : Carl Lemaire

compute canada | calcul canada

Calcul Québec

Partenaires financiers

Plan

Calcul Québec

1. Bibliothèques logicielles
2. Graphe de calcul
3. Matériel
4. Gestion des données
5. Hyperparamètres

Les bibliothèques

Numpy, Scikit-Learn, TensorFlow, PyTorch

Bibliothèques logicielles

Calcul Québec

	Spécialité	Caractéristiques	Matériel
Numpy	Mathématiques numériques	Peut être utilisé pour faire des NN **(don't)**	CPU
Scikit-Learn	ML général	Variété de modèles encapsulés et standardisés (ex. NN)	CPU
TensorFlow (TF) **PyTorch**	Deep Learning	Flexibilité et expressivité pour les NN	CPU GPU TPU

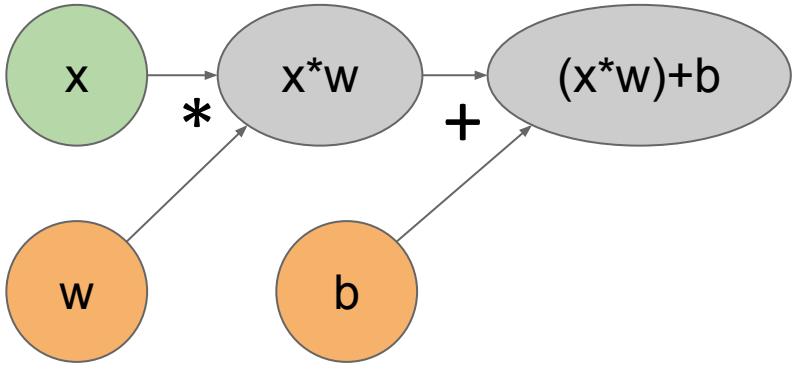
Graphe de calcul

Explicite, implicite, statique, dynamique

Graphe de calcul

Calcul Québec

- NN: graphe d'opérations
- Rétropropagation
- Pourquoi construire le graphe?
  - Différentiation automatique



Construction du graphe

Calcul Québec

Construction explicite (statique)

- TF 1
- “Define-**and**-run”

Construction implicite (dynamique)

- TF 2, PyTorch
- “Define-**by**-run”

Graphe explicite

Calcul Québec

```
model = Sequential()
model.append( FullyConnectedLayer() )
model.append( FullyConnectedLayer() )
# “model” contient le graphe


y = model.predict(x)
```

Graphe implicite

Calcul Québec

```
fc1 = FullyConnectedLayer()
fc2 = FullyConnectedLayer()


y = fc2.forward( fc1.forward( x ))
# “y” contient le résultat ET le graphe
(construit pendant le calcul)
```

Graphe explicite (TF 1)

Calcul Québec

```
in = Input()
fc1, fc2 = FullyConnected(), FullyConnected()
out = fc2( fc1( in ))
# “in” et “out” sont des “placeholders”


model = Model(input=in, output=out)
y = model.predict(x)   # “y” est le résultat
```

Construction du graphe

Calcul Québec

Construction explicite (statique)

- TF 1
- “Define-**and**-run”
- Désavantages
  - Flot de contrôle (boucles, branchages)
  - Déboguage

Construction du graphe

Calcul Québec

Construction implicite (graphe dynamique)

- TF 2, PyTorch
- “Define-**by**-run”
- Désavantages
  - Manipulation du graphe
  - Optimisation, compilation

Dynamique / Statique

Calcul Québec

Flexible
Moins portable

Optimisé
Plus portable

Code (graphe dynamique)	Langage intermédiaire	Graphe statique
`y = torch.matmul(x, w)`	TorchScript scripting	TorchScript tracing
`y = tf.matmul(x, w)`	tf.function + AutoGraph	tf.function tracing

TF v.s. PyTorch: historique

Calcul Québec

- Époque antique (avant 2019)
  - **Compétition dans les approches**
  - TF 1.x
    - explicite seulement
    - orienté production
    - Keras pour simplifier le prototypage
  - PyTorch 0.x
    - implicite seulement
    - orienté prototypage
- Époque contemporaine (depuis 2019)
  - **Convergence des approches**
  - Définition toujours implicite (define-by-run)
  - Tracing et langage intermédiaire

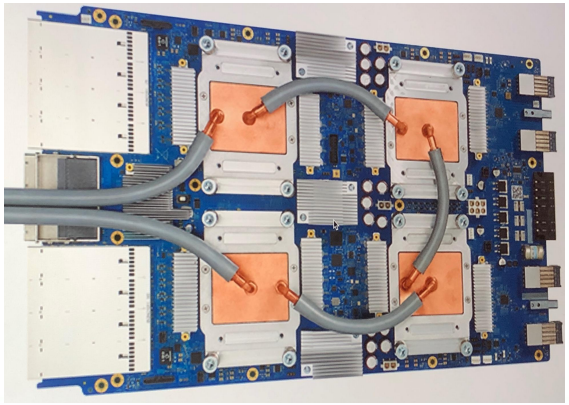
Matériel

Accélérateurs (GPU, TPU)

Calcul Québec

- GPU
  - **Graphics** Processing Unit
  - Spécialisé pour: **Géométrie vectorielle**

- TPU
  - **Tensor** Processing Unit
  - Spécialisé pour: **Deep Learning**
  - ~30x plus de performance/Watt en inférence v.s. GPU (Jouppi et al. 2017)
  - Seulement chez Google Cloud



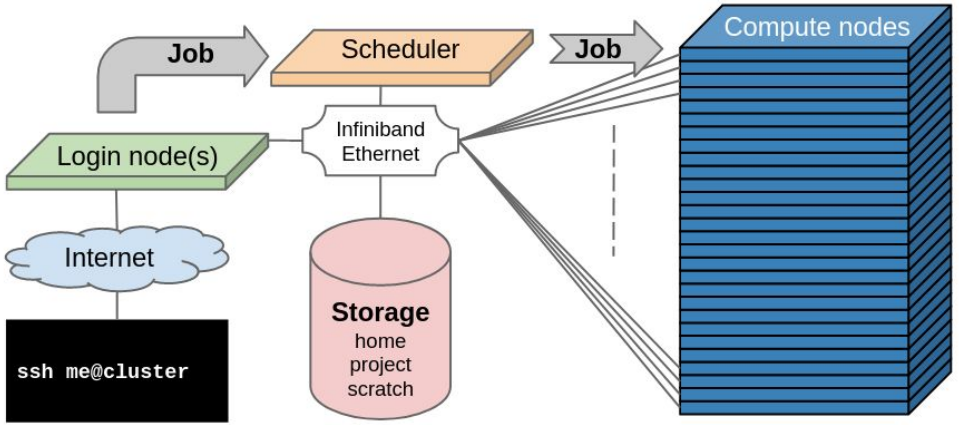
Grappe de calcul

Calcul Québec

a.k.a. “Superordinateur”

- Centaines de noeuds (ordinateurs) gérés par un ordonnanceur (*scheduler*)
- Allocation de ressources
- Soumissions de tâches (*jobs*)
- Stockage partagé
  (Douleur partagée)

À lire: [Using Compute Canada Clusters for ML Research]



Gestion des données

Gestion des données

Calcul Québec

Situation: ImageNet sur stockage partagé

- Dataset avec 1 million de petites images
- Stockage partagé centralisé (grappe)
- Problème: **Accès disque/réseau inefficient**
- Solution: **Encapsuler et transférer au lieu de calcul**
- Exemple: `tar` (archiver sans compresser)

Gestion des données

Calcul Québec

Situation: Énorme base de données, complexe

- Arborescence de données (tenseurs ou autre)
- Utiliser pickle?
- Problème: **Pas possible de tout charger en RAM**
- Solution: **Utiliser HDF5**
- Hierarchical Data Format

Gestion des données

Calcul Québec

	Encapsulation	Hiérarchique	Accès partiel
Dossier de fichiers			
`tar`	✔		
`torch.save` / `pickle`	✔	✔	
HDF5	✔	✔	✔

Hyperparamètres

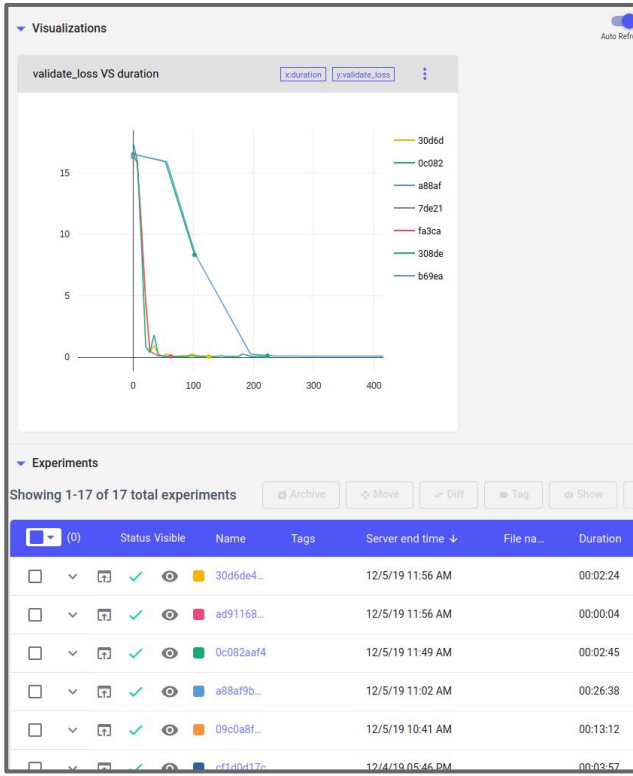
Hyperparamètres

Calcul Québec

- Paramètres: “poids” dans le réseau
- Hyperparamètres: “configuration”
  - Exemple: # couches, *learning rate*, *weight decay*
- Problème d’optimisation en soi!

Hyperparamètres

- Approches
  - Essai-erreur (a.k.a *“l’expérience”*)
  - *Grid search, random search*
  - Algorithmes d’optimisation
- Outils
  - Weights & Biases, Comet.ml
  - Ax
  - Keras tuner



Atelier

Atelier

Calcul Québec

1. Connexion
    a. S’assurer de pouvoir se connecter au serveur
    b. Vous aurez tous un username et mot de passe
2. Transfert des données et du code
    a. Base de données TinyImageNet (tinyimagenet.tar)
    b. Code fourni: github.com/lemairecarl/atelier-dl-cc

Atelier

Calcul Québec

3. Concevoir le script de soumission
    a. Faire fonctionner le code (tâche interactive)
4. Soumettre la tâche
    a. Tâche “batch” (100% automatisée)

Atelier

Calcul Québec

5. Récupérer les sorties de la tâche
    a. Std out/err
    b. Checkpoints, prédictions
6. Suivre le déroulement de la tâche
    a. Tensorboard
    b. % utilisation GPU

Tutoriels

Calcul Québec

Il vous est fortement recommandé de suivre ce tutoriel sur PyTorch:

https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Ces tutoriels sur TensorFlow pourraient aussi vous intéresser:

https://www.tensorflow.org/tutorials/quickstart/beginner

https://www.tensorflow.org/tutorials/keras/classification

https://www.tensorflow.org/guide/gpu