



Bienvenue!

**ÉCOLE D'HIVER FRANCOPHONE
EN APPRENTISSAGE PROFOND**

5 - 9 mars 2018



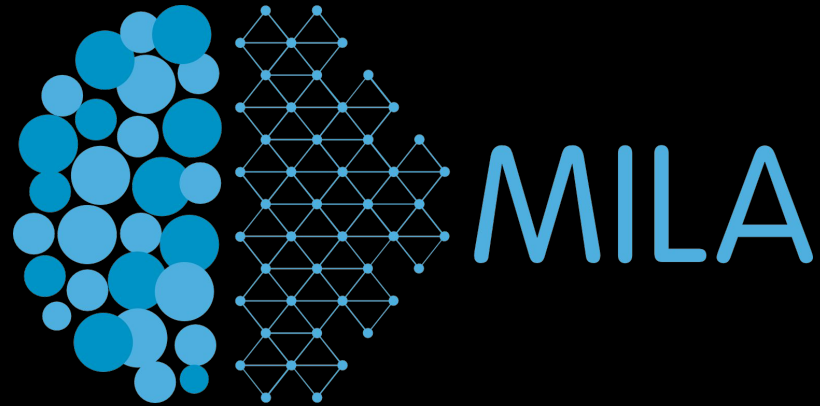
IVADO

HEC Montréal
Polytechnique Montréal
Université de Montréal



MILA

Institut
des algorithmes
d'apprentissage
de Montréal



Conseils pratiques : entraînement et mise en oeuvre informatique

Guillaume Alain

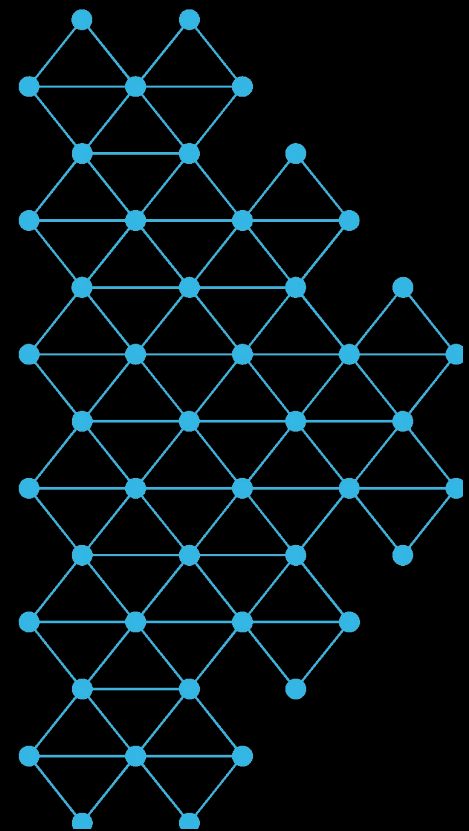
guillaume.alain.umontreal@gmail.com

Spoilers

1. Mettre GPU récent par nVidia dans desktop Linux correct.
2. Possibilité de Amazon EC2, Google Cloud, MS Azure.
3. Utiliser des modèles préfabriqués en ligne.
4. Imiter le code des experts. S'inspirer sur github.
5. Vérifier qu'on obtient des outputs corrects. Sanity checks.
6. Bien organiser ses expériences. Éviter les fouillis.

Bonus : Ensemble de modèles pour classification.

Comment s'équiper pour
faire du Deep Learning ?



Graphical Processing Unit (GPU)

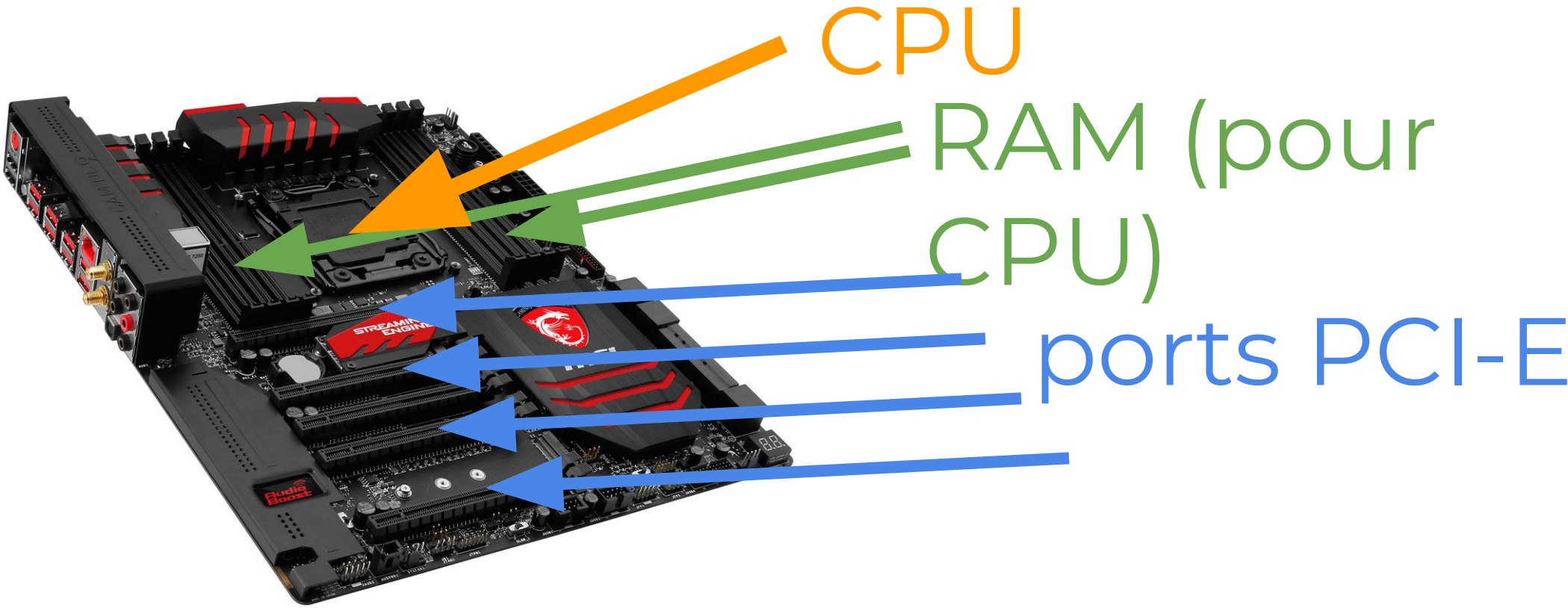
Accélérateur de calcul, avec sa propre mémoire vive (RAM), branché dans le port PCI-E sur la carte mère.

Possibilité de faire beaucoup d'opérations simples en parallèle.

- Faire des calculs sur beaucoup de triangles pour faire du rendu 3D.
- Multiplier des grosses matrices.



Graphical Processing Unit (GPU)



Graphical Processing Unit (GPU)

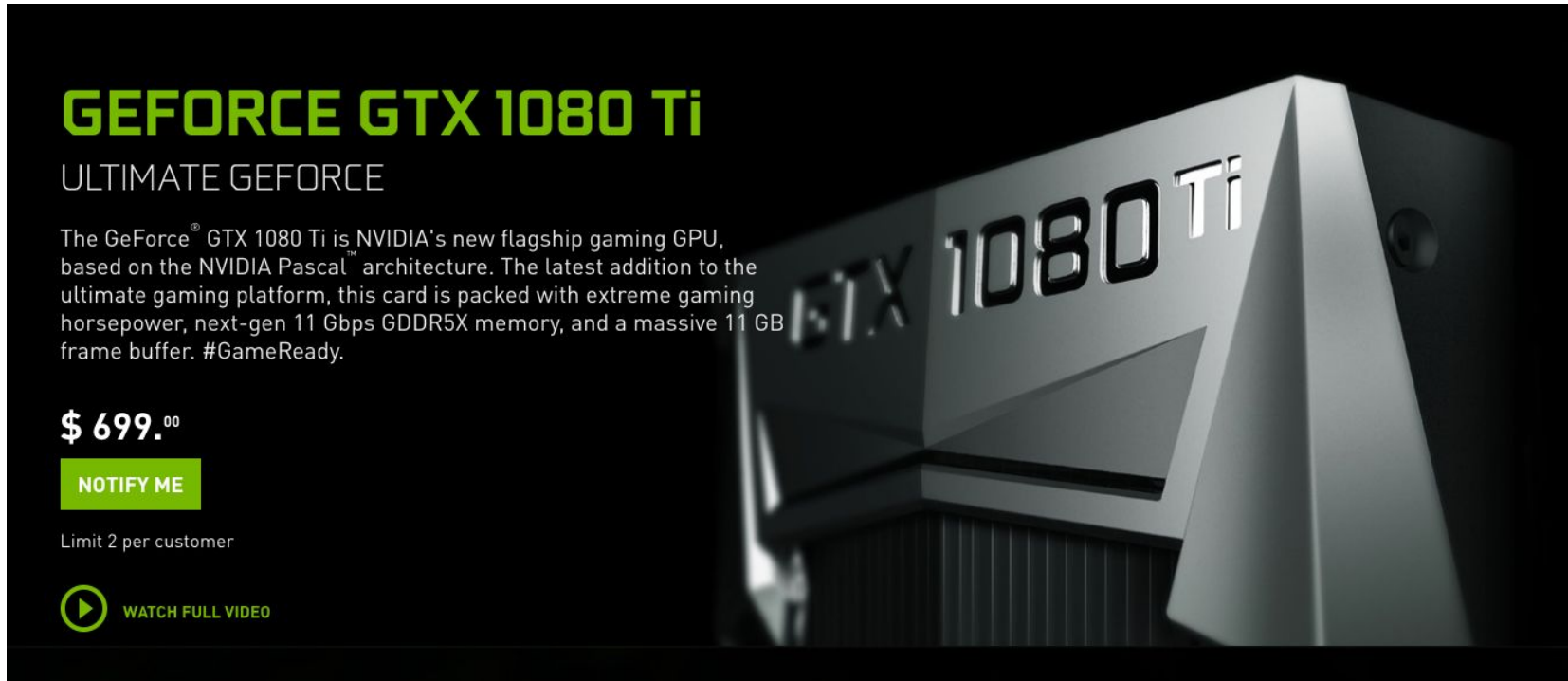


Graphical Processing Unit (GPU)

nVidia ont décidé de faire un pari sur le deep learning en développant des bibliothèques pour accélérer les opérations utiles pour les gens qui font de l'apprentissage profond.



Graphical Processing Unit (GPU)

An advertisement for the NVIDIA GeForce GTX 1080 Ti GPU. The background is a dark, stylized image of the GPU with 'GTX 1080 Ti' written on its side. The text is in a clean, modern font. The price is prominently displayed in white. A green button for 'NOTIFY ME' and a play button for 'WATCH FULL VIDEO' are also present.

GEFORCE GTX 1080 Ti

ULTIMATE GEFORCE

The GeForce® GTX 1080 Ti is NVIDIA's new flagship gaming GPU, based on the NVIDIA Pascal™ architecture. The latest addition to the ultimate gaming platform, this card is packed with extreme gaming horsepower, next-gen 11 Gbps GDDR5X memory, and a massive 11 GB frame buffer. #GameReady.

\$ 699.⁰⁰

NOTIFY ME

Limit 2 per customer

▶ WATCH FULL VIDEO

Graphical Processing Unit (GPU)

NVIDIA TITAN Xp

\$ 1,200.⁰⁰

ADD TO CART

Free Shipping

Limit 2 per customer



Graphical Processing Unit (GPU)

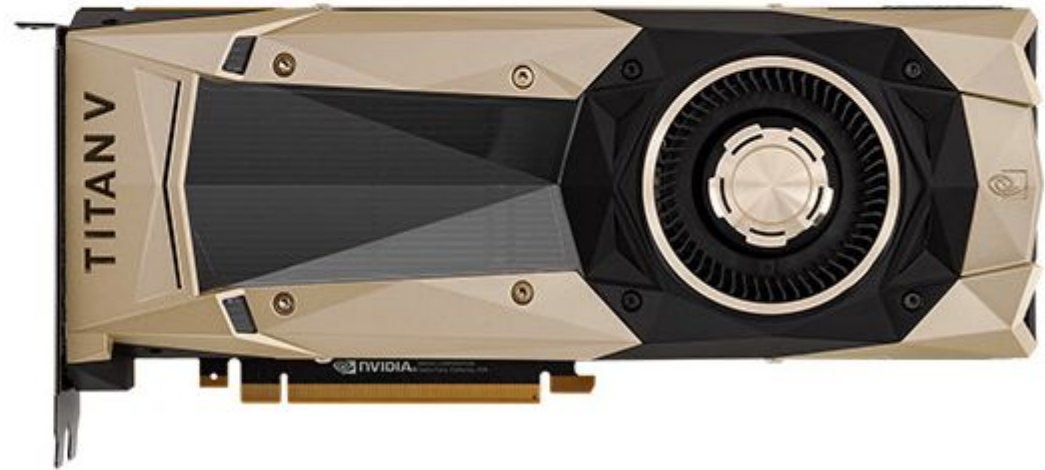
NVIDIA TITAN V

\$ 2,999.⁰⁰

ADD TO CART

Free Shipping

Limit 2 per customer



Graphical Processing Unit (GPU)

GPU pour “gamers”

GTX 1050 Ti (~350 CAD)

GTX 1080 Ti (~1500 CAD)

C'est quoi l'objectif ?

- se tremper l'orteil dans le Deep Learning ?
- déployer un système de production avec vraiment beaucoup de calcul sur des GPU ?

Quelques solutions suggérées

La solution minimale correcte

- GTX 1080 Ti dans un desktop
- bon disque dur SSD
- disque dur d'extra avec quelques TB
- Linux (de préférence)
- un bon processeur quadcore (ex : i7)
- pas besoin de monstre Xeon 16 cores (argent mal dépensé)
- 32 GB de RAM (ou possiblement 64 GB, quoique pas prioritaire)

(Possibilité de mettre deux GPUs au besoin.)

Source de courant



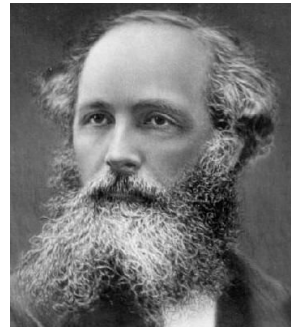
Une GTX Titan ça tire environ 250W.
Deux Titans + CPU ça donne environ 600-700W.

Il faut prévoir une marge de manoeuvre pour éviter d'avoir des ennuis. Personnellement j'utilise une Corsair RM1000 (~200\$) pour deux gros GPUs et ça marche bien.

Mettre 4 GPUs dans une machine, c'est pas si difficile, mais ça demande une expertise que je n'ai pas.

Génération de GPU

Fermi < Kepler < Maxwell < Pascal < Volta



I won an itunes gift card at work and don't want it.

I'll take it!
I usually buy from there.



lol_comics.livejournal.com



J'ai des crédits AWS qui me sont complètement inutiles

Je vais les prendre.
J'entraîne mes modèles profonds sur AWS.





- K80 (okay) pour 0.45 USD/h
- P100 (wow!) pour 1.46 USD/h

- Ça ne vaut pas la peine pour essayer un GPU pour faire du Deep Learning. Autant payer 200\$ et mettre un GPU dans son desktop.
- Vraiment plus facile que d'installer son propre cluster pour utiliser 10+ GPUs.

Performance CPU sans GPU

Personnellement, pour des tâches de classification d'images j'ai pu observer une différence d'environ 80x entre l'entraînement sur CPU et sur GPU.

C'est la différence entre

- ça roule pendant un mois
- ça roule cette nuit et c'est prêt demain matin

Sur certains modèles de NLP, la différence peut être d'un facteur de 2-4x. Les benchmarks les plus menteurs pourraient dire 200x dans les meilleurs cas. Disons que 20x c'est peut-être plus raisonnable.

Performance CPU sans GPU

Entraînement sur dataset de 1 million d'exemples.
On veut faire 10 passes sur le dataset au complet, à
raison de 100 exemples par seconde (avec GPU).

	Entraînement	Évaluation sur 1000 images
GPU	28 heures	10 secondes
	2216 heures (92 jours)	80 secondes (une minute)

Hardware Apple

Il n'existe pas de laptop Apple récent avec un GPU nVidia. Pas de desktop Apple non plus.

Les laptops Dell XPS ont une GTX1050.

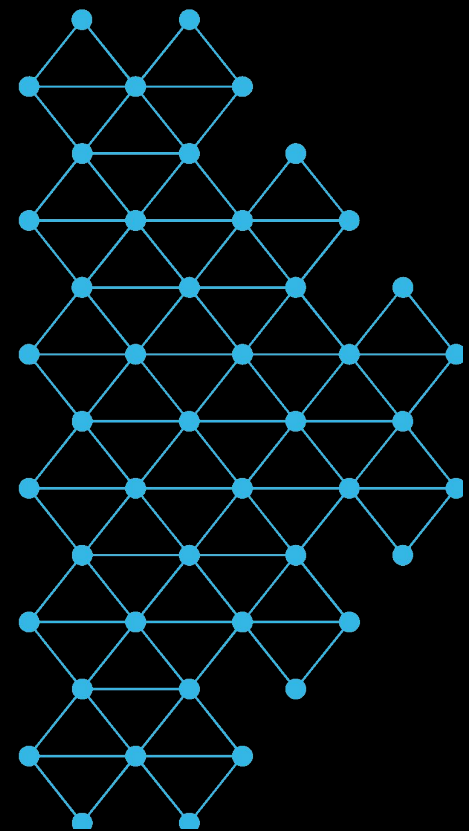


Goulot d'étranglement

Il faut que le goulot d'étranglement ("bottleneck") soit l'aspect computationnel du modèle d'apprentissage profond.

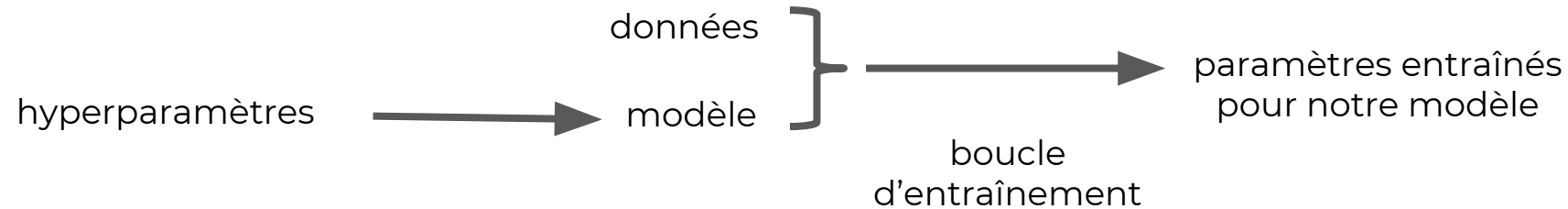
- Pas le chargement des données sur le disque dur.
- Pas la mémoire disponible pour stocker le modèle désiré sur GPU.

Taille minibatch	Temps pour une minibatch	Total exemples / seconde	Coût en mémoire
8	300 ms	~ 25	100 MB + 1000 MB
16	300 ms	~ 50	100 MB + 2000 MB
32	300 ms	~ 100	100 MB + 4000 MB
64	600 ms	~ 100	100 MB + 8000 MB

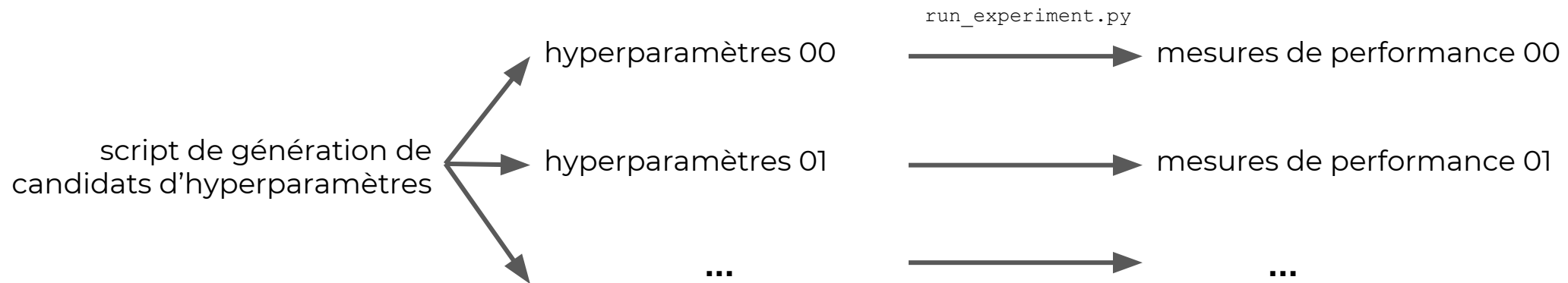


Rouler une expérience de DL

diagramme d'entraînement pour une instance de modèle

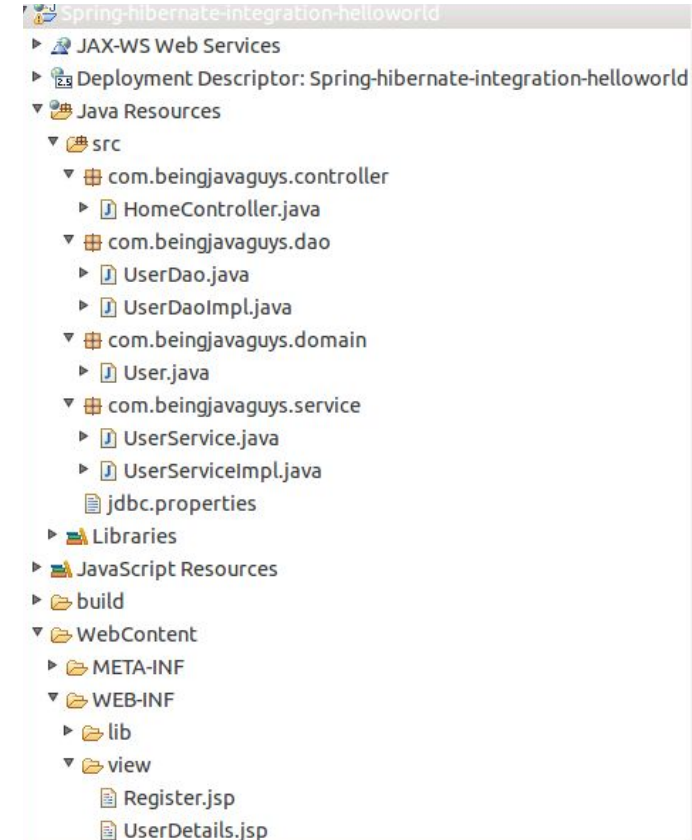
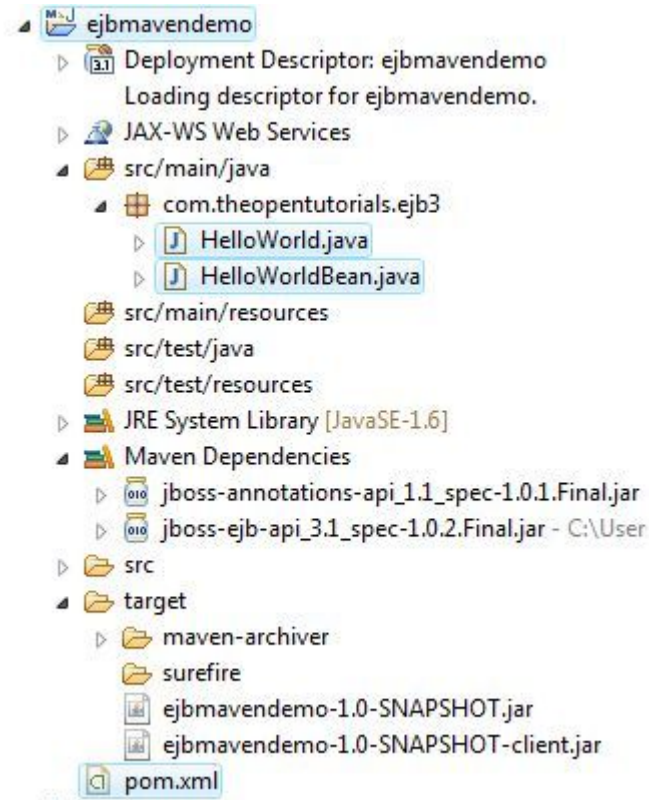


exploration d'hyperparamètres pour obtenir le meilleur modèle





Quel est l'équivalent "deep learning" de ça ?



A Research to Engineering Workflow

by Dustin Tran

<http://dustintran.com/blog/a-research-to-engineering-workflow>

```
-- doc/
  -- 2017-nips/
    -- preamble/
    -- img/
    -- main.pdf
    -- main.tex
    -- introduction.tex
-- etc/
  -- 2017-03-25-whiteboard.jpg
  -- 2017-04-03-whiteboard.jpg
  -- 2017-04-06-dustin-comments.md
  -- 2017-04-08-dave-comments.pdf
-- src/
  -- checkpoints/
  -- codebase/
  -- log/
  -- out/
  -- script1.py
  -- script2.py
-- README.md
```

```
.
├── .gitignore
├── README
├── data
│   └── download.sh
├── hpsearch
│   ├── hyperband.py
│   └── randomsearch.py
├── main.py
├── models
│   ├── __init__.py
│   └── basic_model.py
├── results
│   └── .gitkeep
└── tests
```

TensorFlow: A proposal of good practices for files, folders and models architecture

<https://blog.metaflow.fr/tensorflow-a-proposal-of-good-practices-for-files-folders-and-models-architecture-f23171501ae3>



Conseil général :

**Voir ce que les autres personnes font
et s'en servir comme inspiration.
Développer son propre style à partir de ça.**

Aller aux conférences de DL/ML et potiner avec les chercheurs.

mise en oeuvre du modèle d'apprentissage profond

Réutiliser autant que possible des modèles déjà disponibles en ligne. Chercher pour “model zoo”.

Parfois on veut même utiliser un modèle pré-entraîné et s'éviter beaucoup de travail. Chercher pour “pretrained model” ou “pretrained model weights”.

mise en oeuvre du modèle d'apprentissage profond



github.com/tensorflow/models

tensorflow / models

Watch 1,338 Unstar 17,665 Fork 6,951

Code Issues 212 Pull requests 37 Projects 0 Wiki Insights

Models built with TensorFlow

886 commits 1 branch 0 releases 207 contributors Apache-2.0

Branch: master New pull request

Create new file Upload files Find file Clone or download

alexnet.py

alexnet_test.py

cifarnet.py

inception.py

inception_resnet_v2.py

inception_resnet_v2_test.py

inception_utils.py

inception_v1.py

inception_v1_test.py

inception_v2.py

inception_v2_test.py

inception_v3.py

inception_v3_test.py

inception_v4.py

inception_v4_test.py

lenet.py

mobilenet_v1.md

mobilenet_v1.png

mobilenet_v1.py

mobilenet_v1_test.py

nets_factory.py

nets_factory_test.py

overfeat.py

overfeat_test.py

resnet_utils.py

resnet_v1.py

resnet_v1_test.py

resnet_v2.py

resnet_v2_test.py

vgg.py

vgg_test.py



mise en oeuvre du modèle d'apprentissage profond



Keras

keras.io/applications

fchollet / keras

Watch 1,140 Star 16,911 Fork 6,029

Code Issues 1,145 Pull requests 33 Projects 1 Wiki Insights

Deep Learning library for Python. Runs on TensorFlow, Theano, or CNTK. <http://keras.io/>

deep-learning tensorflow theano neural-networks machine-learning data-science

3,687 commits 5 branches 30 releases 480 contributors

Branch: master New pull request

Create new file Upload files Find file Clone or download

Classify ImageNet classes with ResNet50

```
from keras.applications.resnet50 import ResNet50
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model = ResNet50(weights='imagenet')

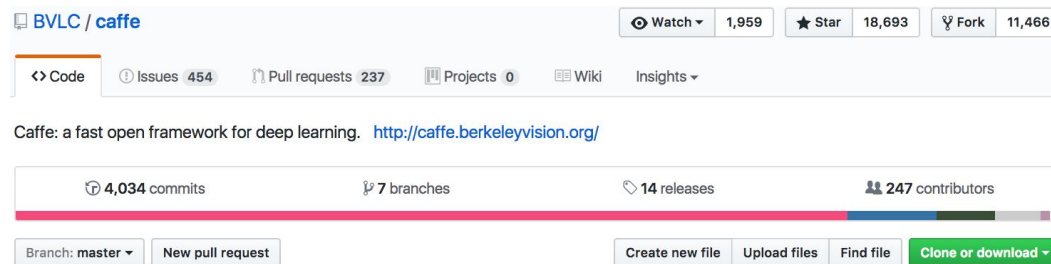
img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
# Predicted: [(u'n02504013', u'Indian_elephant', 0.82658225), (u'n01871265', u'
```

mise en oeuvre du modèle d'apprentissage profond

Caffe

github.com/BVLC/caffe/wiki/Model-Zoo



BVLC / caffe

Watch 1,959 Star 18,693 Fork 11,466

Code Issues 454 Pull requests 237 Projects 0 Wiki Insights

Caffe: a fast open framework for deep learning. <http://caffe.berkeleyvision.org/>

4,034 commits 7 branches 14 releases 247 contributors

Branch: master New pull request

Create new file Upload files Find file Clone or download

ResNets: Deep Residual Networks from MSRA at ImageNet and COCO 2015

This repository contains the original models (ResNet-50, ResNet-101, and ResNet-152) described in the paper "Deep Residual Learning for Image Recognition" (<http://arxiv.org/abs/1512.03385>). These models are those used in [ILSVRC] (<http://image-net.org/challenges/LSVRC/2015/>) and COCO 2015 competitions, which won the 1st places in: ImageNet classification, ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

More instructions with prototxt and binary weight files are in: <https://github.com/KaimingHe/deep-residual-networks>

Reference:

```
@article{He2015,
  author = {Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun},
  title = {Deep Residual Learning for Image Recognition},
  journal = {arXiv preprint arXiv:1512.03385},
  year = {2015}
}
```

mise en oeuvre du modèle d'apprentissage profond

```
from resnet50 import ResNet50
from keras.preprocessing import image
from imagenet_utils import preprocess_input,
decode_predictions

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
print('Predicted:', decode_predictions(preds))
# print: [[u'n02504458', u'African_elephant']]
```

mise en oeuvre du modèle d'apprentissage profond

```
from resnet50 import ResNet50
from keras.preprocessing import image
from imagenet_utils import preprocess_input,
decode_predictions

model = ResNet50(weights='imagenet')

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
print('Predicted:', decode_predictions(preds))
# print: [[u'n02504458', u'African_elephant']]
```

```
def preprocess_input(x, data_format=None):

    if data_format == 'channels_first':
        # 'RGB'->'BGR'
        x = x[:, ::-1, :, :]
        # Zero-center by mean pixel
        x[:, 0, :, :] -= 103.939
        x[:, 1, :, :] -= 116.779
        x[:, 2, :, :] -= 123.68
    else:
        # 'RGB'->'BGR'
        x = x[:, :, :, ::-1]
        # Zero-center by mean pixel
        x[:, :, :, 0] -= 103.939
        x[:, :, :, 1] -= 116.779
        x[:, :, :, 2] -= 123.68
    return x
```

mise en oeuvre du modèle d'apprentissage profond

Machine Learning: The High-Interest Credit Card of Technical Debt

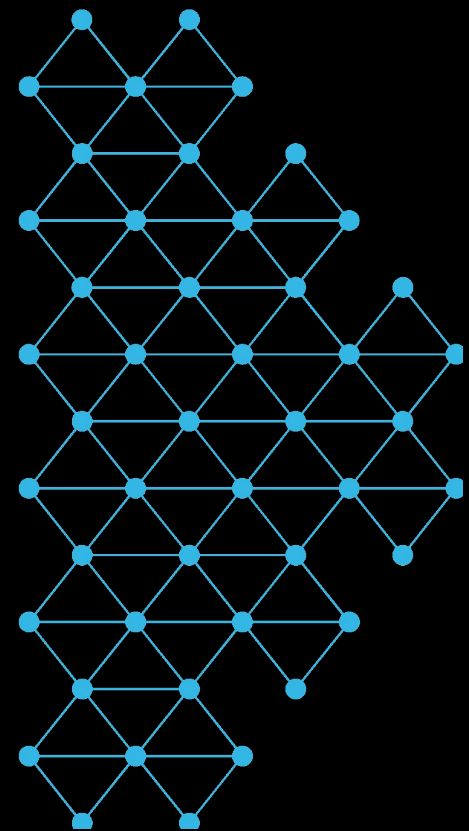
**D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young**
{dsculley, gholt, dgg, edavydov}@google.com
{toddpillips, ebner, vchaudhary, mwyoung}@google.com
Google, Inc

Abstract

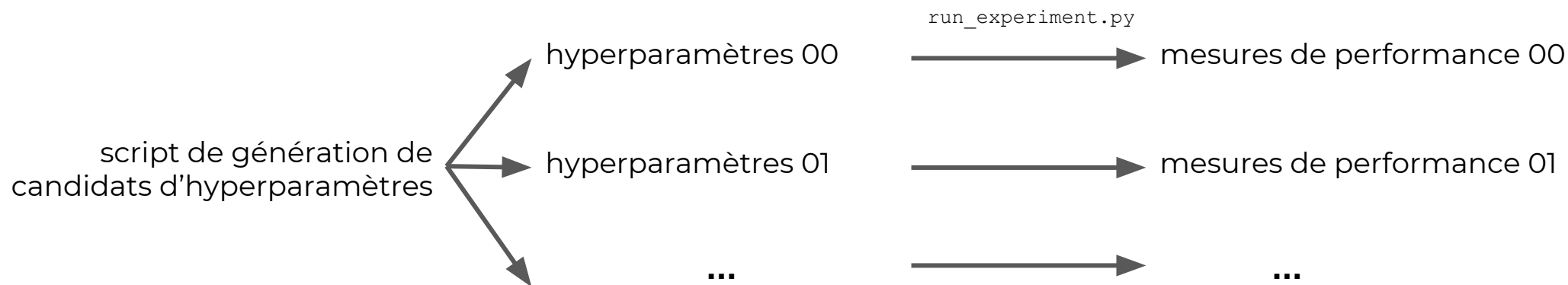
Machine learning offers a fantastically powerful toolkit for building complex systems quickly. This paper argues that it is dangerous to think of these quick wins as coming for free. Using the framework of *technical debt*, we note that it is remarkably easy to incur massive ongoing maintenance costs at the system level when applying machine learning. The goal of this paper is highlight several machine learning specific risk factors and design patterns to be avoided or refactored where possible. These include boundary erosion, entanglement, hidden feedback loops, undeclared consumers, data dependencies, changes in the external world, and a variety of system-level anti-patterns.



Exploration d'hyperparamètres



exploration d'hyperparamètres pour obtenir le meilleur modèle



Chaque expérience doit produire assez de logs pour

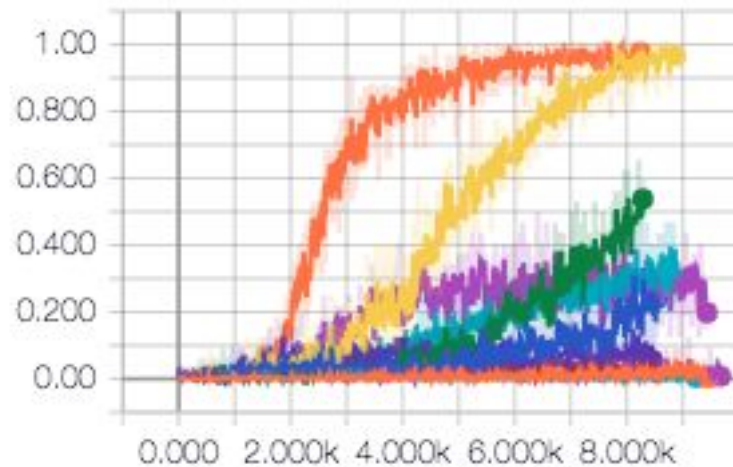
- diagnostiquer les erreurs et les comportements louches
- permettre de sélectionner celle qu'on considère comme la plus propice à mieux généraliser à des données nouvelles

Tensorboard vient avec Tensorflow. Pour s'en servir, il faut utiliser les "summaries" en Tensorflow.

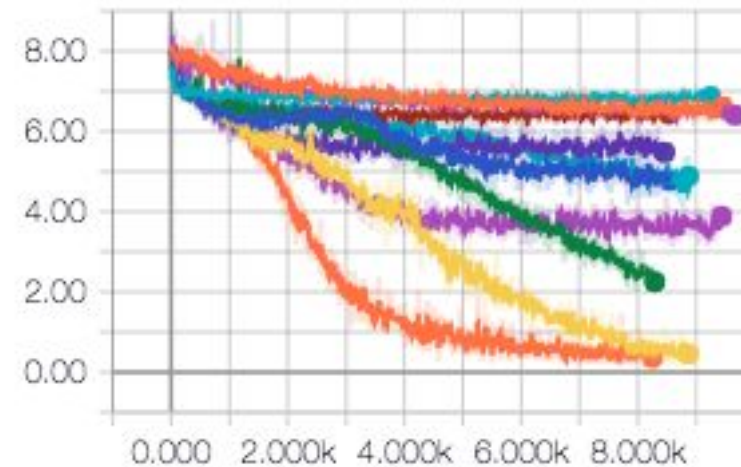
```
batch_loss = tf.nn.softmax_cross_entropy_with_logits(...)
loss = tf.reduce_mean(batch_loss)
batch_accuracy = ...
accuracy = tf.reduce_mean(tf.cast(batch_accuracy, "float"))

tf.summary.scalar("train/loss", loss)
tf.summary.scalar("train/accuracy", accuracy)
```

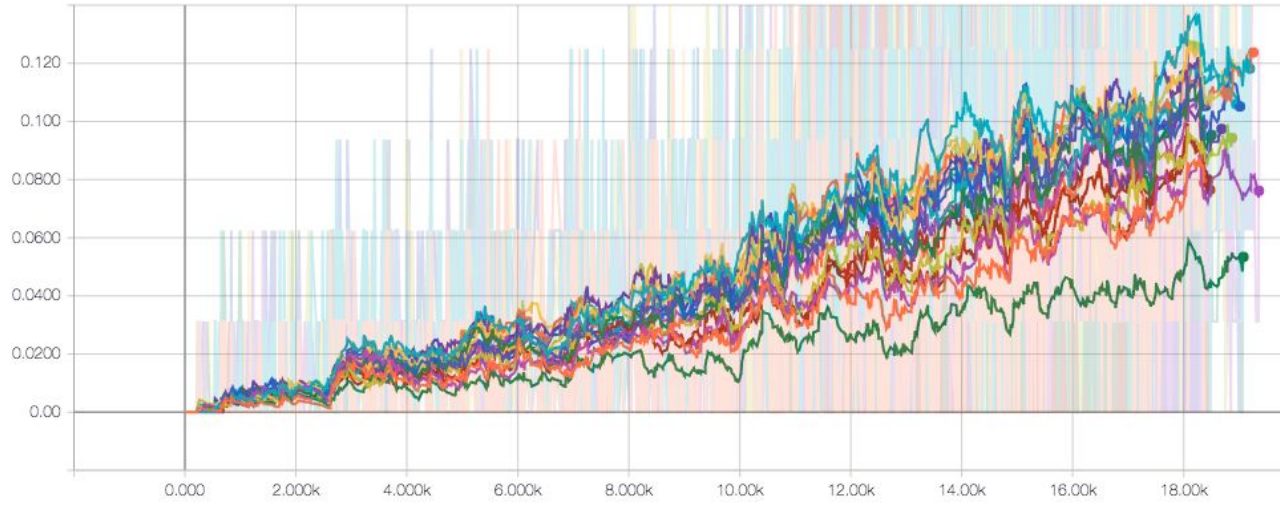
model/train/accuracy



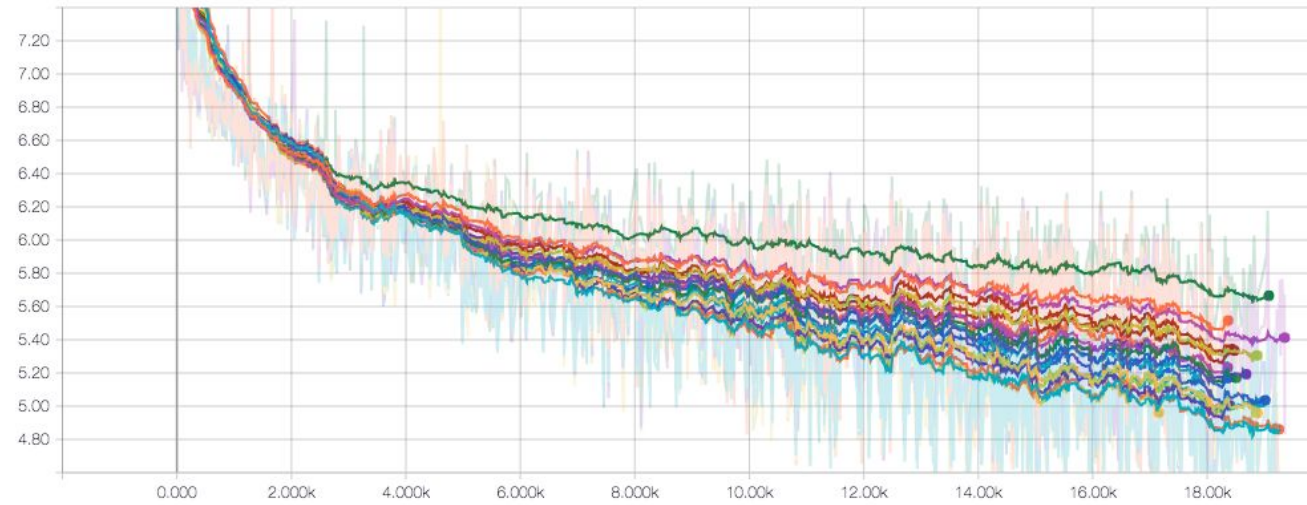
model/train/loss



model/train/accuracy



model/train/loss



Conseil général :

**Quand on roule plusieurs expériences à grande échelle,
il faut être capable de générer de manière indépendante
les graphiques qui affichent les résultats.**

Une expérience est lancée

```
python performing_training.py --expdir="experiment_000"
```

et après 10 heures de calculs elle génère un graphique

```
experiment_000/loss_over_training_epochs.png
```

Conseil général :

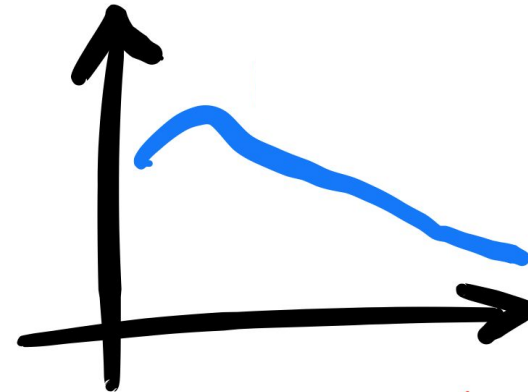
**Quand on roule plusieurs expériences à grande échelle,
il faut être capable de générer de manière indépendante
les graphiques qui affichent les résultats.**

Une expérience est lancée

```
python performing_training.py --expdir="experiment_000"
```

et après 10 heures de calculs elle génère un graphique

```
experiment_000/loss_over_training_epochs.png
```



Conseil général :

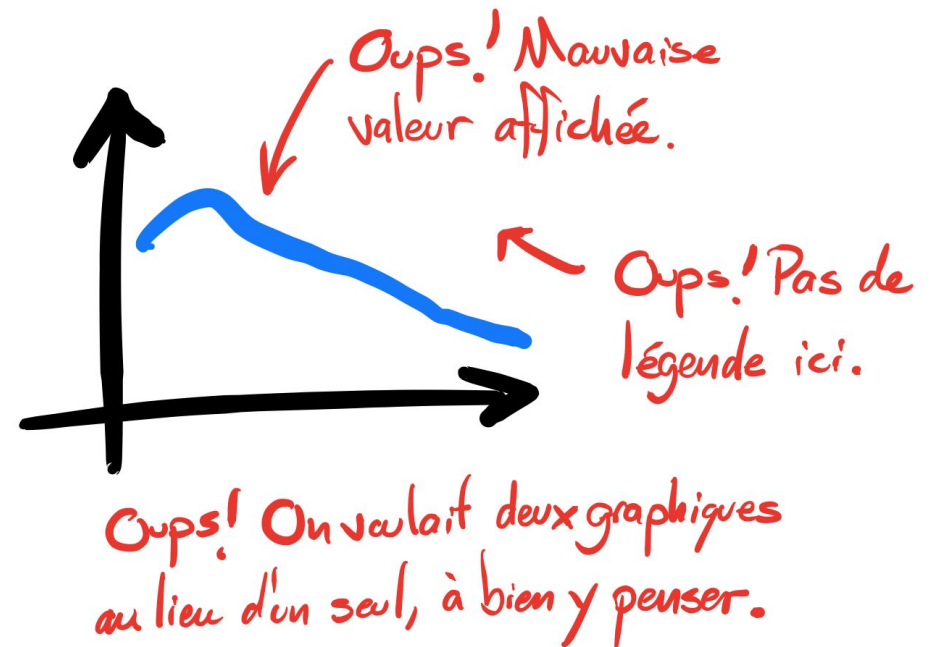
**Quand on roule plusieurs expériences à grande échelle,
il faut être capable de générer de manière indépendante
les graphiques qui affichent les résultats.**

Une expérience est lancée

```
python performing_training.py --expdir="experiment_000"
```

et après 10 heures de calculs elle génère un graphique

```
experiment_000/loss_over_training_epochs.png
```



Conseil général :

Une expérience qui peut être interrompue en plein milieu puis relancée sans être affectée par l'interruption, c'est beaucoup plus facile à gérer.

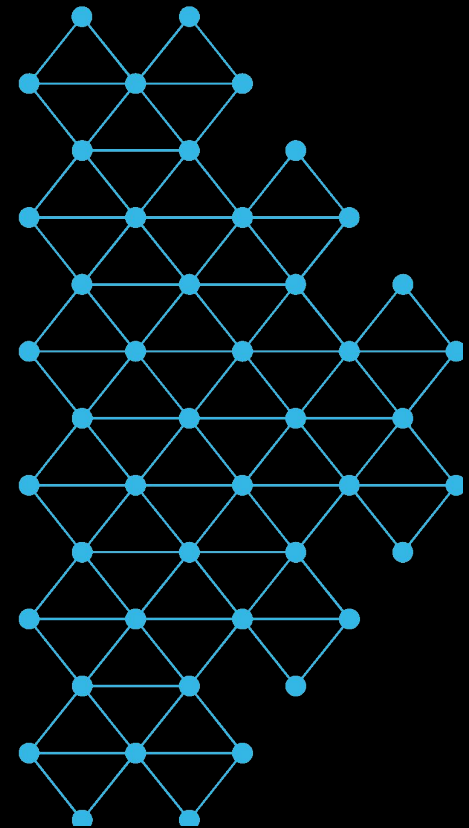
```
with tf.Session(  
    config=tf.ConfigProto(  
        log_device_placement=False,  
        allow_soft_placement=True)  
    ) as session:  
  
    # Your code for the main loop here.
```

Pas facile à interrompre.

```
sv = tf.train.Supervisor(  
    logdir=hparams.train_dir,  
    init_op=init_op,  
    summary_op=merged_summaries,  
    saver=saver,  
    global_step=global_step,  
    save_model_secs=600)  
  
with sv.managed_session(  
    config=tf.ConfigProto(  
        log_device_placement=False,  
        allow_soft_placement=True)  
    ) as session :  
  
    # Your code for the main loop here.
```

Continuable
automatiquement.

Ensemble de modèles



Ensemble de modèles

Votre méthode expérimentale :

données



classification
30% d'erreur

Ensemble de modèles

Votre méthode expérimentale :

données



classification
30% d'erreur

La méthode populaire :

données



classification
25% d'erreur

Ensemble de modèles



Ensemble de modèles



On fait un vote populaire à chaque fois.

Pour se tromper, il faut que 2 ou 3 des experts se trompent en même temps.

$$P(\text{erreur}) = 0.7*0.3*0.3 + 0.3*0.7*0.3 + 0.3*0.3*0.7 \\ + 0.3*0.3*0.3$$

$$P(\text{erreur}) = 0.216$$

Les chances d'erreur sont maintenant de 21.6 %. On est passés de 30% à 21.6% juste en combinant 3 instances du même modèle !?

WTF !?

Ensemble de modèles



30% erreur



25% erreur



21.6% erreur

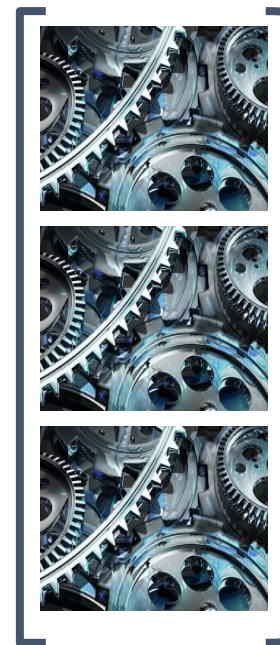
Ensemble de modèles



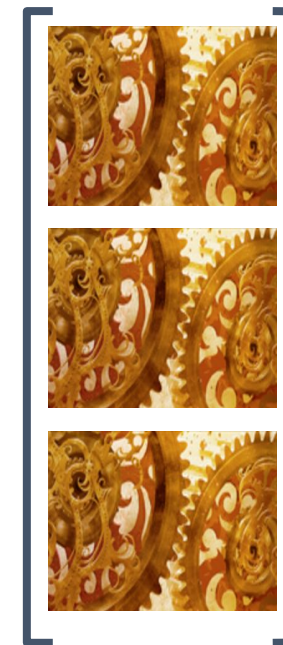
30% erreur



25% erreur



21.6% erreur



15.6% erreur

Ensemble de modèles



25% erreur

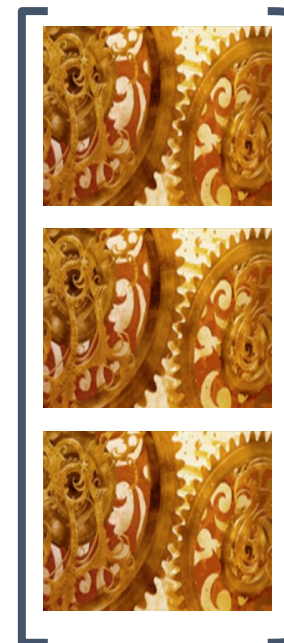


15.6% erreur

Ensemble d'experts



25% erreur



15.6% erreur

Ensemble de modèles



Le raisonnement s'applique uniquement si les instances de modèles font chacun une classification de manière **indépendante**.

Un même modèle entraîné 10 fois va avoir tendance à faire des erreurs un peu différentes, mais pas si radicalement différentes.

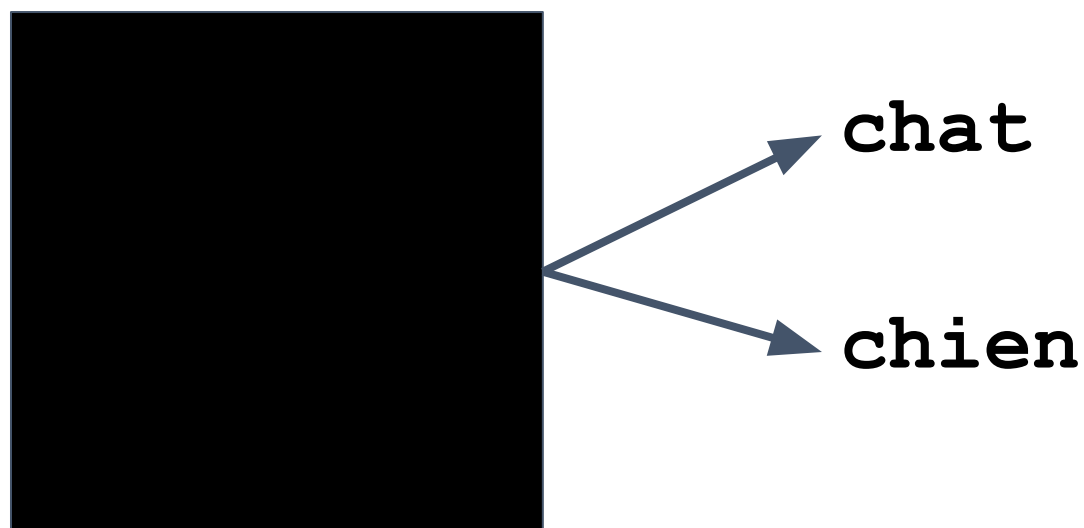
Ensemble de modèles

Un groupe d'experts trop pareils,
ça se trompe ensemble de manière coordonnée.



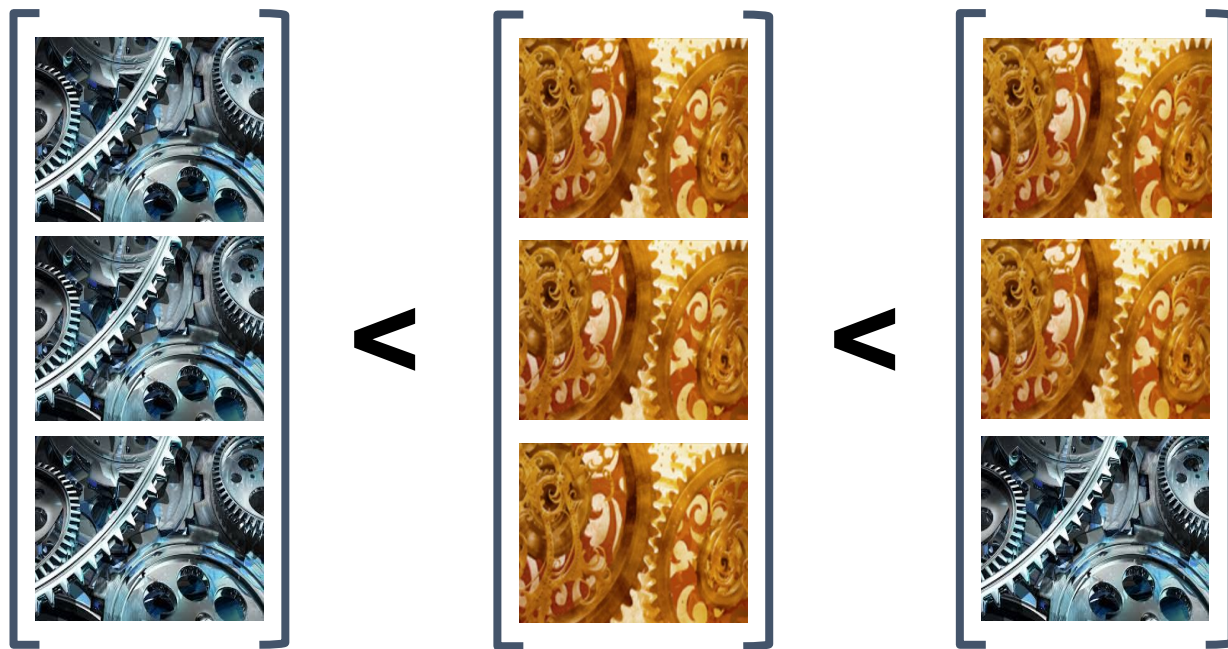
Ensemble de modèles

Peu importe le nombre d'experts qu'on met ensemble, on n'aura jamais une meilleure classification sur certains exemples ambigus.



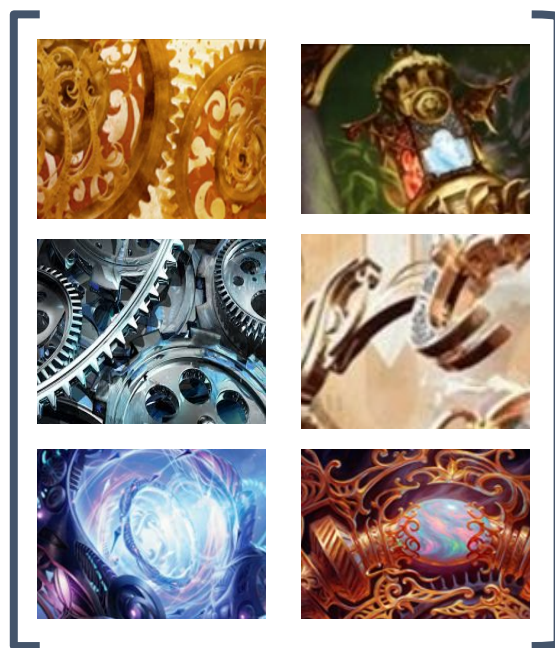
Ensemble de modèles

Il se peut très bien qu'on ait une situation comme ça :



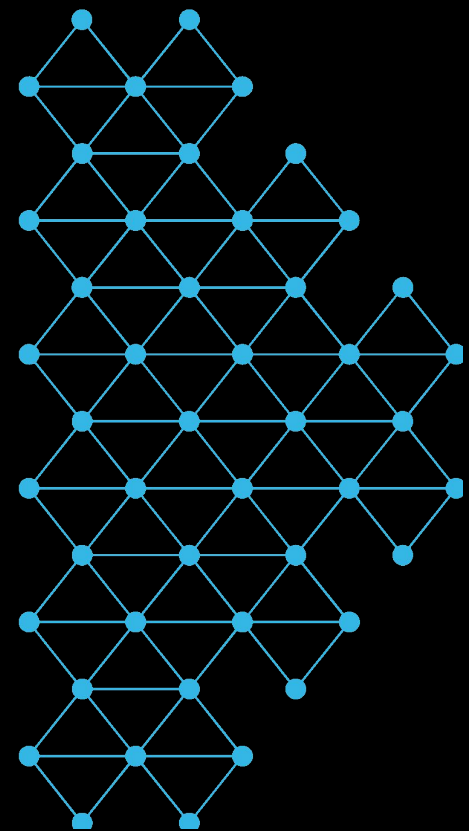
Ensemble de modèles

Penser au coût computationnel de rouler plusieurs modèles en parallèle. Possibilité de prendre des modèles moins chers pour mitiger ça.



Ensemble de modèles





Merci !
Bonne aventure dans le
merveilleux monde
du **Deep Learning** !

Contact

<http://mila.umontreal.ca/mila-fr/>

Guillaume Alain guillaume.alain.umontreal@gmail.com



1. Mettre GPU récent par nVidia dans desktop Linux correct.
2. Possibilité de Amazon EC2, Google Cloud, MS Azure.
3. Utiliser des modèles préfabriqués en ligne.
4. Imiter le code des experts. S'inspirer sur github.
5. Vérifier qu'on obtient des outputs corrects. Sanity checks.
6. Bien organiser ses expériences. Éviter les fouillis.

Bonus : Ensemble de modèles pour classification.

Guillaume Alain guillaume.alain.umontreal@gmail.com