RAPPORT DE STAGE

Norvège

2ème année F4

Projet réalisé par

Julien Feuillas

le 16 mai 2019

État de l'art

Tuteur de Stage : **Arvid Lundervold**
Co-encadrant de Stage : **Alexander Lundervold**

Jury

Arvid Lundervold,	Professeur UiB	Maître de Stage
Murielle Mouzat,	Professeure ISIMA	Communication
Vinvent Barra,	Professeur ISIMA	Tuteur ISIMA

durée : 5 mois

Isima 1 rue de la Chebarde, Aubière 63170

Partie 1

Deep Learning

1.1 Qu'est-ce que c'est ?

Il s'agit d'une variante du Machine Learning. Le Machine Learning permet de déterminer des relations au sein d'un jeu de données et ainsi de penser un modèle mathématiques qui permettra par la suite de retrouver sans trop de calcul où placer une donnée que nous voudrions ajouter à ce jeu de données.

Le Deep Learning (aussi appelé en français l'apprentisage profond), détermine des modèles mathématiques plus complet. Les relations peuvent alors être plus complexe. Les principales méthodes utilisées pour réaliser du Deep Learning sont les réseaux de neurones.

1.2 Langage

Les méthodes de Machine Learning (ainsi que celles de Deep Learning) sont possibles pour beaucoup de langages de programmation :

— C
— C++
— R
— Java

Mais le langage le plus utilisés dans ce domaine au niveau de la recherche est le Python. Ce langage interprété est très utilisé et populaire dans le domaine de la recherche scientifique, c'est pourquoi les différentes avancées en termes de Machine Learning et Deep Learning se sont faites avec ce langage.

1.3 Évolution

Au cours des dernières années, l'intérêt pour les méthodes de Machine Learning en général a été de plus en plus important ce qui a permis d'accélérer la recherche dans le domaine des réseaux de neurones. La plupart des projets de recherche en termes de machine learning étant open-source, cela permet également à cette dernière de faire un bon en avant.

Ainsi on a pu rapidement voir arriver des méthodes pour traiter des problèmes comme de l'imagerie ou des données médicales.

Partie 2

TensorFlow

Cette librairie Python a été développée dans le but de faciliter la prise en main des réseaux de neurones dans le cadre du Deep Learning. Actuellement la dernière version "stable" est la 1.13.1, mais la 2.0.0 est en cours de test.

Il existe 2 types d'installation pour TensorFlow. La première est la plus simple à mettre en place, mais elle demendera plus de temps à réaliser une itération lors de l'exécution du code. Cette première installation se fait sur le CPU (Le microprocesseur) de l'ordinateur. La deuxième installation se fait par contre sur le GPU (la carte graphique). Cela implique que l'ordinateur doit être pourvu d'un GPU compatible pour pouvoir réaliser cette installation.

2.1 Fonction de perte

Les réseaux de neuirones de TensorFlow s'appuient sur un système de fonction de perte pour l'entraînement. Ainsi, à chaque itération, nous calculons le résultat de la fonction de perte. Le but étant de minimiser les pertes, c'est à dire de minimser le résultat de cette fonction. Pour cela, des fonctions d'optimisation ont été mises en place au sein de TensorFlow.

2.2 Optimisation

2.2.1 Descente de gradients

Il s'agit d'une méthode assez connue dans le domaine de l'optimisation. Elle peut par contre poser problème si la fonction que nous essayons d'optimiser n'est pas strictement convexe. Il suffit qu'il y ait plus d'un minimum local pour nous bloquer et nous empêcher de trouver le minimum absolu (s'il est présent).

2.2.2 Adam

Cette méthode a été développé lorsque l'intérêt pour le Machine Learning et plus particulièrement pour le Deep Learning s'est développé. Elle est à ce jour, la méthode la plus populaire lorsqu'il s'agit d'optimiser des fonctions de pertes dans le cadre du Deep Learning [2].

2.3 Le taux d'apprentisage

Ce paramètre permet de déterminer le pas utilisé pour optimiser la fonction de perte. Plus ce paramètre sera important et plus le pas utilisés sera important. Ainsi, dans les réseaux de neurones classiques, nous retrouverons le plus souvent un taux d'apprentissage constant. Le problème qui peut alors se poser est si la fonction se retrouve dans un puits et que pour en sortir il est nécessaire d'avoir un plus grand pas.

Il faut par contre faire attention à ne pas avoir un taux d'apprentissage trop élevé no plus parce que nous avons aussi la possibilité de manquer le minimum absolu de la fonction.

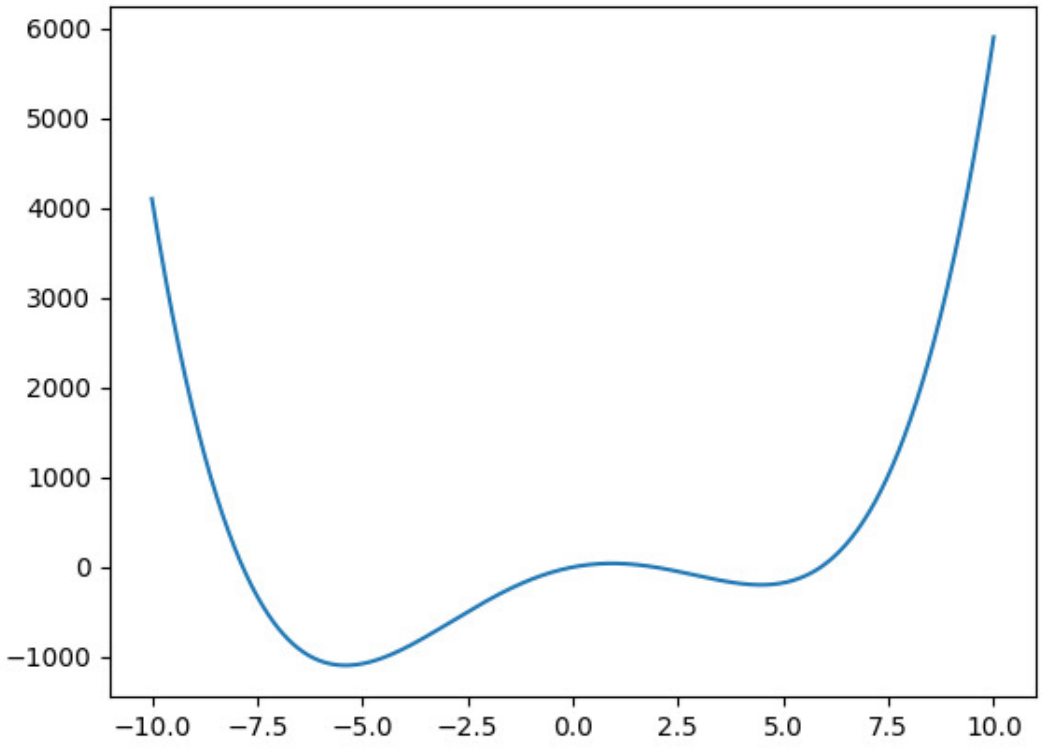

FIGURE 2.1 – pas = 1, abscisse actuelle : 5.0

Il est aussi possible de faire varier le taux d'apprentissage [3]. De cette manière, avec un taux d'apprentissage variant de manière périodique, nous avons la possiblité de passer un gap trop important en augmentant le taux d'apprentissage, mais cela nous permet aussi de ne pas rater le puits recherché (lorsque le taux d'apprentissage diminue). La variation que [3] trouve la plus adaptée est celle-ci :

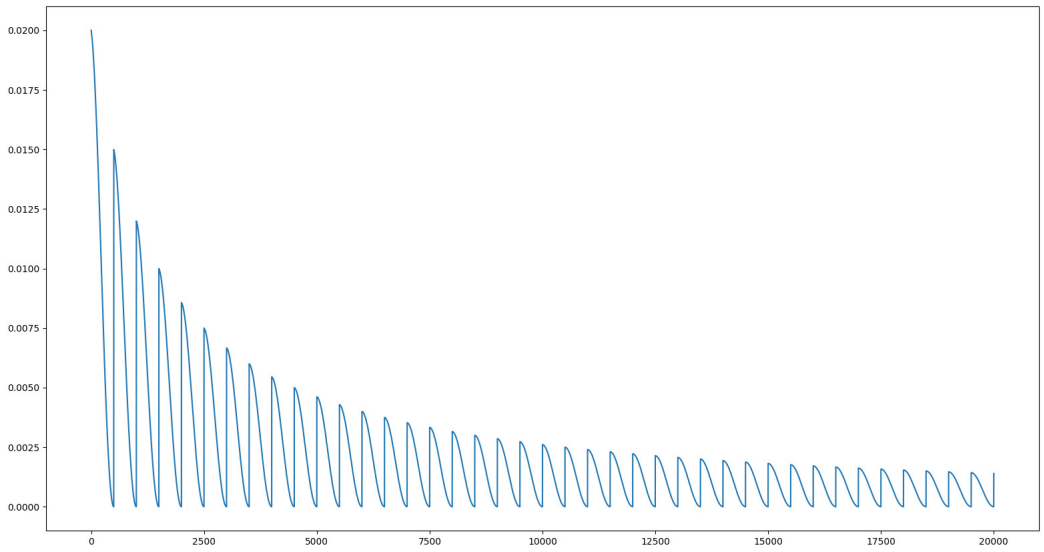

FIGURE 2.2 – Modification du taux d'apprentissage à chaque itération

Partie 3

NiftyNet

NiftyNet est un projet open-source en Python qui permet de réaliser des réseaux de neurones en Deep Learning sans écrire une seule ligne de Python. Il permet à l'aide d'un fichier de configuration (fichier INI) d'entraîner des réseaux de neurones et de réaliser des prédictions après cet entraînement.

Ce projet a été réalisé spécifiquement pour le domaine médical, et plus particulièrement pour le domaine de l'imagerie médicale.

NiftyNet s'appuie énormément sur TensorFlow pour fonctionner.

Bibliographie

[1] Tensorflow. https://www.tensorflow.org/,
date of consultation : April 2019.

[2] Vitaly Bushaev. Adam – latest trends in deep learning optimization, October 2018. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c,
date of consultation : 25th April 2019.

[3] Vitaly Bushaev. Improving the way we work with learning rate, November 2018. https://techburst.io/improving-the-way-we-work-with-learning-rate-5e99554f163b,
date of consultation : 30th April 2019.

[4] Eli Gibson, Wenqi Li, Carole Sudre, Lucas Fidon, Dzhoshkun I. Shakir, Guotai Wang, Zach Eaton-Rosen, Robert Gray, Tom Doel, Yipeng Hu, Tom Whyntie, Parashkev Nachev, Marc Modat, Dean C. Barratt, Sébastien Ourselin, M. Jorge Cardoso, and Tom Vercauteren. Niftynet : a deep-learning platform for medical imaging. Computer Methods and Programs in Biomedicine, 2018.
date of consultation : April 2019.

[5] Eli Gibson, Wenqi Li, Carole Sudre, Lucas Fidon, Dzhoshkun I. Shakir, Guotai Wang, Zach Eaton-Rosen, Robert Gray, Tom Doel, Yipeng Hu, Tom Whyntie, Parashkev Nachev, Marc Modat, Dean C. Barratt, Sébastien Ourselin, M. Jorge Cardoso, and Tom Vercauteren. Niftynet source code, April 2019. https://github.com/NifTK/NiftyNet,
date of last consultation : 25th April 2019.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[7] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. https://www.sciencedirect.com/science/article/pii/S0939388918301181, page 26, December 2018.

[8] Marc Modat, Miklos Espak, Eli Gibson, Imanol Luengo, Dzhoshkun Shakir, Zach Eaton-Rosen, Carole Sudre, Tom Vercauteren, Matteo Mancini, Guotai Wang, Lucas Fidon, Wenq Li, Jorge Cardoso, Matt Clarkson, Mian Asbat Ahmad, and Tom Doel. Niftynet, October 2018. https://cmiclab.cs.ucl.ac.uk/CMIC/NiftyNet,
date of consultation : 4th April 2019.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in python, March 2019. https://scikit-learn.org/stable/.

[10] Wikipedia. Content-based image retrieval, March 2019.
https://en.wikipedia.org/wiki/Content-based_image_retrieval,
date of consultation : 3rd April 2019.