BERT Loris
PHAM Duy Anh Philippe
VATIN Clément

Quantum Computing
CPE Lyon
05/2020

TENSORFLOW QUANTUM

Project Report

Résumé

TensorFlow Quantum (TFQ) est une nouvelle bibliothèque de Machine Learning Quantique développée par Google pour le prototypage rapide de modèles de Machine Learning hybrides quantique-classique.

Le but de ce projet est de réaliser un tutoriel sur TensorFlow Quantum sous la forme d'un Jupyter Notebook et d'implémenter un réseau de neurone quantique (QNN).

GitHub : https://github.com/LorisBERT/QuantumComputingProject

Google Collab : https://colab.research.google.com/drive/1yYL4UUY6Af9qVJYIoa2uGyeeO3-ESaMZ?usp=sharing

Table des matières

Introduction

Dans ce travail nous nous sommes attardés sur la création d'un réseau de neurones quantique (Quantum Neural Networks ou QNN). L'objectif est de résoudre un problème de classification à partir de la base de données d'images de chiffres manuscrits MNIST. Nous comparons par la suite le résultat du QNN avec celui d'un réseau de neurones classique équivalent. Nous aurons alors un benchmark des deux approches pour cette application précise.

Contexte

La faible puissance des processeurs quantiques actuels ne nous permet pas encore d'implémenter un réseau de neurones quantique directement sur une puce physique (via *IBM Quantum Experience* par exemple). Pour cette raison, nous allons nous contenter d'un processeur quantique simulé à partir d'un processeur classique tout le long de ce projet. Nous allons également adapter le QNN afin de travailler sur une base de données plus restreinte, toujours dans un souci de performance. L'objectif consiste donc à classifier des images de 0 et de 1, de taille 4 par 4 pixels.

Preprocessing

La base de données MNIST est composées de 70.000 images de chiffres manuscrits allant de 0 jusqu'à 9 et de dimension 28 par 28 pixels. Nous devons donc dans un premier temps extraire les images qui nous intéressent puis dans un souci de performance pour l'algorithme quantique, puis les binariser avant l'entrainement (figure 1).

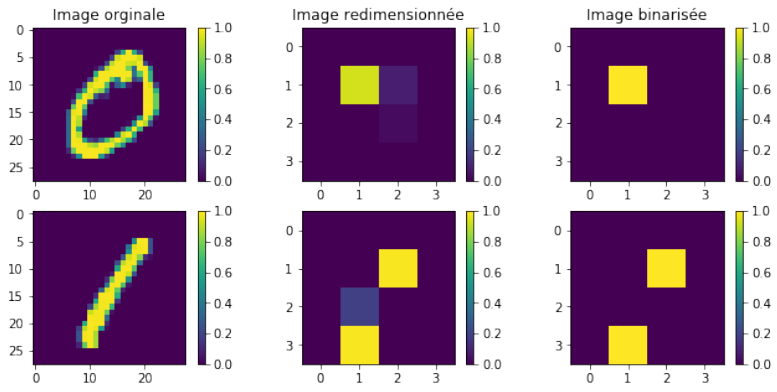

Figure 1
Préparation des images pour l'entrainement

Il est important de noter que sur les 70.000 images du MNIST, 60.000 images sont utilisées pour l'entrainement et 10.000 images sont réservées pour l'évaluation du réseau de neurones. Après extraction nous avons un set de 14.780 images contenants uniquement des 0 et 1. Nous avons pour le set d'entrainement 12665 images, ce qui représente 85.7% du set extrait et 2115 images pour celui d'évaluation soit 14.3%. Les mêmes proportions et la répartition statistique du dataset MNIST sont conservés par construction de celui-ci (équiprobabilité des classes).

Pour le traitement quantique, l'image doit être ré-encodée : chaque pixel est ainsi caractérisé sur un qubit dont l'état de départ dépend de la valeur du pixel correspondant. Cela explique l'intérêt de la binarisation qui nous permet de n'utiliser qu'un seul qubit par pixel et de fait le même nombre de qubits par images.

Création du modèle Quantique

Le réseau de neurones quantique est constitué de 2 layers, chacun implémentant n instances de la même porte à 2 qubits (avec n la tailles des images, 16 dans notre cas). Chaque qubits de donnée est relié à un qubit de lecture et permet ainsi à l'information de se propager le long du modèle.

Portes	Nombre
XX	16
ZZ	16

La fonction de coût choisie est *Hinge* (définie par $L(y) = \max(0, 1 - y \cdot \hat{y})$) car très bien adaptée aux problèmes de classification tel que celui-ci. Elle est associée à l'optimisateur *Adaptative Momentum estimation* (Adam) qui a pour but d'accélérer le processus de descente de gradient.

Création du modèle Classique

Le réseau de neurones quantique est constitué de 2 Dense layers avec un nombre total de paramètres similaire au QNN afin de garantir une certaine équité entre les deux modèles.

Layers	Taille	Paramètres
Input	16	0
Dense	2	34
Dense	1	3

La fonction de coût et l'optimisateur sont les mêmes que pour le QNN (*Hinge* et *Adam* respectivement).

Résultats

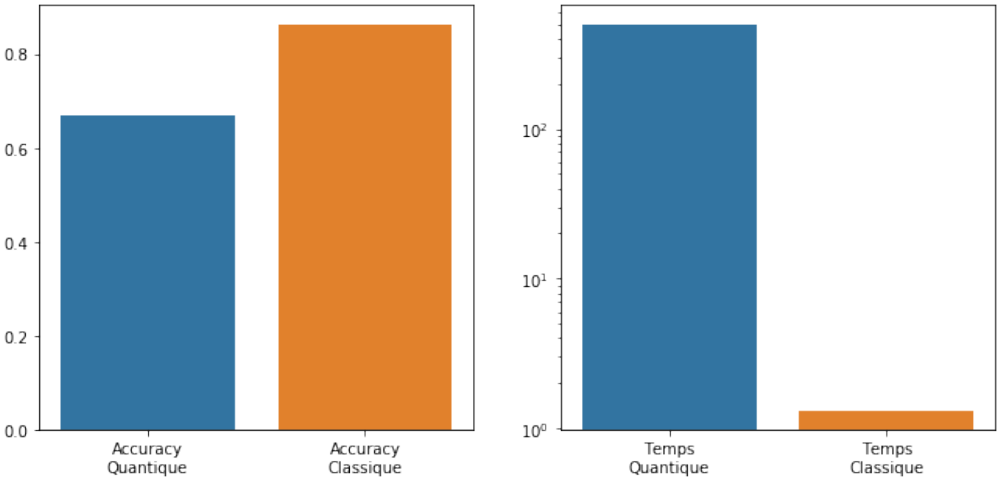

Figure 2
Comparatif des performances entre les deux réseaux de neurones

La différence la plus notable entre ces deux réseaux de neurones est leur temps d'exécution. En effet, le modèle classique à été entrainé avec un sous ensemble de 1024 images sur 10 epochs en quelques secondes alors que le modèle quantique n'a été entrainé que sur 3 epochs et a pris plus de 8 minutes. Le modèle classique est donc 1500 fois plus rapide à entrainer, et ce, sans utiliser de cartes graphiques qui peuvent encore diviser le temps de calcul par 10 ou plus.

Malgré ce temps d'entrainement plus long, le QNN n'est pas meilleur pour autant. Il parvient à atteindre un score de plus de 65% avec seulement 1024 images de très basse résolution ce qui est honorable, mais se fait dépasser par le réseau classique qui atteint les 90%. Avec l'ensemble du dataset, le modèle est toutefois capable d'atteindre les 85%, mais reste inférieur à son homologue classique.

Conclusion

Pour conclure, même si cela reste possible, les réseaux de neurones entièrement quantiques ne sont pas adaptés à ce genre de taches. La meilleure solution reste une approche hybride où le processeur quantique se contente de calculer la fonction de coût avant de retourner les informations au processeur classique qui fait les ajustements nécessaires puis appelle une nouvelle fois le processeur quantique et ainsi de suite.

"For classical data, it is difficult to beat a classical neural network" (Documentation de TFQ)

Remerciements

Nous remercions notre enseignant M. Bruno Fedrici, pour son enseignement ainsi que les pistes et documentations fournies afin de nous guider dans ce projet.

Références principales

[1] Dawid Kopczyk, *Quantum machine learning for data scientists* (04/2018)

[2] Michael Broughton et al., *TensorFlow Quantum: A Software Framework for Quantum Machine Learning* (03/2020)

[3] Zhih-Ahn Jia et al., *Quantum Neural Network States: A Brief Review of Methods and Applications* (02/2019)

[4] Kerstin Beer et al., *Training deep quantum neural networks* (2020)

[5] https://www.tensorflow.org/quantum

[6] https://github.com/tensorflow/quantum

[7] https://www.tensorflow.org/

[8] https://github.com/tensorflow

[9] https://cirq.readthedocs.io/

[10] https://github.com/quantumlib/cirq