Rapport initial :

Apprentissage automatique avec tinyML dans les réseaux logiciels

Contexte :

Le machine learning est un domaine d'étude de l'intelligence artificielle, qui repose sur l'apprentissage semi-autonome d'un programme via un modèle mathématique. Grâce à de larges quantités de données pré-traitées (données d'entraînement), le logiciel apprend à reconnaître une corrélation entre ces données pour déterminer un résultat le plus juste possible sur des données qui n'ont pas encore été analysées (données de test ou réelles). Ce domaine a eu un grand gain de popularité dans les dernières années pour 3 raisons :

- Premièrement, la mise en circulation de nombreux appareils informatiques et capteurs, notamment avec l'IoT (IoT = Internet of Things, la connexion à internet de nombreux objets dotés de capteurs), est à l'origine d'une très grande quantité de données à analyser pour des nombreuses raisons : analyse du comportement, prévisions météo, etc…
- Des processeurs spécialisés aux calculs nécessaires au machine learning ont nettement progressé (GPU et TPU = Graphics/tensor processing unit).
- Des algorithmes de plus en plus performants et efficaces ont vu le jour.

L'importance croissante des appareils IoT dans divers domaines (santé, automobile et les dispositifs portables) rend essentiel le développement de modèles de Machine Learning efficaces sur des plateformes limitées par leur batterie et leur performance. Cependant, le gain de popularité du Machine Learning tel qu'il existe principalement (centralisé) présente ses limites :

Utiliser les capteurs de l'IoT pour analyser leurs données à distance implique une latence élevée, tandis que l'utilisation des processeurs spécialisés pour la performance nécessite une quantité d'énergie phénoménale, et entraîne un coût de production supplémentaire. Une solution à ces deux problèmes est l'*edge computing*: le traitement non centralisé des données, notamment en local sur les différents systèmes IoT. Cela implique de réaliser du machine learning sur des ordinateurs très peu puissants, rarement équipés de processeurs spécialisés. Cela représente le TinyML, une forme de Machine learning en plein essor et marquant

une convergence entre l'IoT et le Machine Learning : effectuer des calculs de Machine Learning sur des microprocesseurs à la consommation extrêmement basse.

Le concept de TinyML a été inventé par des chercheurs cherchant à intégrer des modèles de Machine Learning sur des microcontrôleurs à faible puissance dès le début des années 2010. Depuis son invention, des avancées significatives ont été réalisées dans l'optimisation des algorithmes, la réduction de la taille des modèles et l'adaptation aux contraintes matérielles spécifiques des microcontrôleurs. Le TinyML utilise notamment la compression de modèle, la quantification, et l'optimisation des hyperparamètres pour des performances optimales sur nos systèmes.

Différents modèles sont utilisés dans le TinyML :

- les réseaux de neurones profonds (DNN)
- les réseaux de neurones convolutionnels (CNN)
- les machines à vecteurs de support (SVM)
- les méthodes d'apprentissage non supervisées comme le clustering (a priori non abordé dans ce projet)

Pourtant, des lacunes dans ce domaine existent et comprennent la nécessité de développer des méthodes de validation et de débogage spécifiques au TinyML, ainsi que l'exploration de nouvelles architectures de modèle adaptées aux contraintes matérielles.

Le TinyML offre donc des opportunités prometteuses pour l'expansion du machine learning dans le cadre de l’IoT. L’enjeu du projet est ainsi de faire fonctionner un algorithme de machine learning le plus efficacement possible sur un micro contrôleur (a priori un raspberry pi), et de le faire interagir en réseau avec d’autres instances du modèle. Notre stratégie est donc assez claire : nous allons tester différents algorithmes de machine learning pour comprendre leur efficacité (définie ci-dessous dans les tâches), et en complexifiant peu à peu les exercices et les datas à analyser et classifier.

Tâches :

- Installer un environnement de test sur un container Docker, sous Linux avec Genann installé.
- Comprendre le fonctionnement de Genann
    - Analyser les fonctions à utiliser dans la librairie, leur fonctionnement.
- Produire un premier modèle capable de classifier MNIST (chiffres manuscrits):
    - Implémenter un réseau de neurones perceptron multicouche dans Genann en déduisant son efficacité par:
        - le temps de training,
        - le temps d'inférence
        - la précision du modèle (Une matrice de confusion sur un échantillon de test, accuracy ratio, ROC curve[1], recall, F1-score…)
        - la consommation de mémoire, du cpu
        - Comparaison de différentes configurations de modèles
        - Mettre à jour / corriger l'UML
- Implémenter un réseau de neurones convolutionnels dans Genann :
    - Comprendre le fonctionnement d'un réseau de neurones convolutionnel
    - Analyser plus en détail le fonctionnement de la librairie pour pouvoir la modifier
    - Réaliser une implémentation d'un réseau de neurones convolutionnels à l'aide d'un réseau de neurones perceptron multicouche.
- Potentiels optimisations possibles :
    - Utilisation de type flottant plus petit (float ou __float16 au lieu de double) ou utilisation de quantification (des poids et/ou des fonctions d'activation)
    - Mise en place du *pruning,* ce qui consiste à éliminer les poids négligeables du modèle
    - Éventuellement influer sur les données d'entrées : réduire leur taille, leur résolution (bien noter l'évolution des performances !)
    - Distillation du modèle pour réduire sa taille

[1] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Listes de tests :

- Test disponible directement sur github avec iris (voir si tout a bien été téléchargé, si tout fonctionne comme prévu)
- Test MNIST classification des numéros écrits à la main, réussir à entraîner le modèle et vérifier la qualité de l'entraînement en menant des tests (mesurer les performances : mémoire, cpu, rapidité, précision)
- Base de données supplémentaires : ImageNet
- Test d'image plus complexes (chiens vs chats ou autres)
- Test impliquant d'autres types de données (exemple : reconnaissance vocale)

Diagrammes :

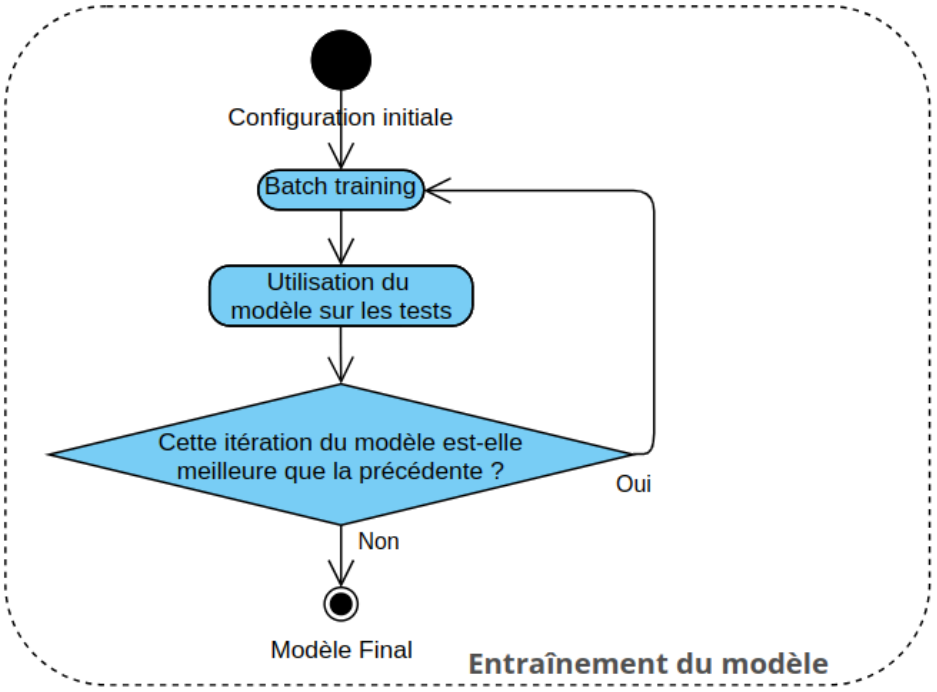
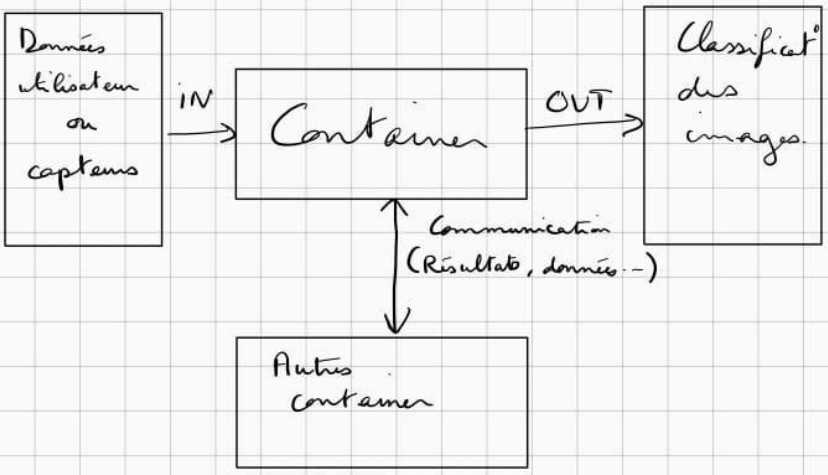
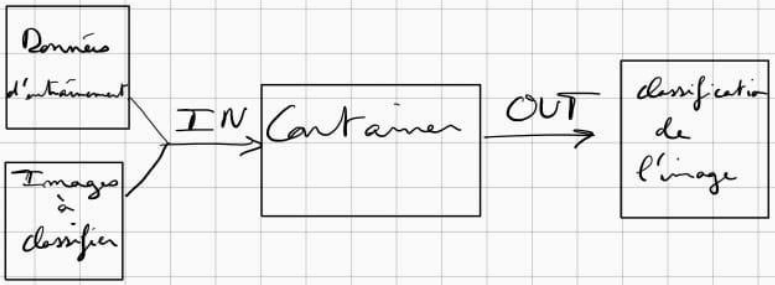


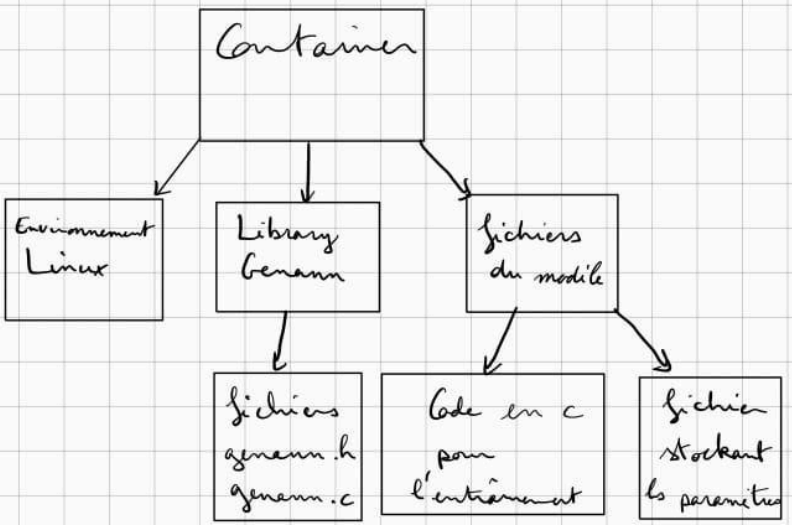
Diagramme composants du système:

Idé globale :



Dans un premier temps :





Rendus dans un premier temps :

- Container tout prêt pour effectuer une classification d'images après entraînement
- Add-on de Genann implémentant un réseau de neurones convolutionnel.

Timeline :

Première partie :

Début mars -> Fin-mars : Implémentation d'un modèle de classification d'image

- Étude de la librairie GenANN (*Tout le monde*)
- Mise en place du Docker (*Edouard*)
- Implémentation du test MNIST sur un premier modèle (*Daniel* + *Roann*)
- Implémentation du calcul des indicateurs de performance (*Antoine*)

Début Avril -> Mi-Mai : Implémentation d'un add-on de CNN dans GenANN

- Apprentissage de la théorie des réseaux de neurones convolutionnels (*Tout le monde*)
- Implémentation de la convolution (*Tout le monde, le travail à réaliser réparti entre nous sera détaillé début avril*)
- Premiers tests de CNN (*convolutional neural network*) (*idem*)

Seconde partie : *La répartition des tâches se fera après détermination des tâches précises à effectuer à partir des résultats de la première partie.*

Mi-Mai -> Fin-Mai :

- Mise en place des différentes optimisations, éventuellement en chercher d'autres
- Mesure de leur utilité

Début Juin -> Mi-Juin :

- Mise en place de la communication réseau pour transférer des données d'entrainements et notifier au robot quelles données utiliser

Mi-Juin -> 28 Juin (17 semaines après le début) (avec la semaine de piscine) :

- Création des rendus finaux, finalisation des optimisations et de la communication

Sources :

1. Partha Pratim Ray,
   A review on TinyML: State-of-the-art and prospects,
   Journal of King Saud University - Computer and Information Sciences,
   Volume 34, Issue 4,
   2022,
   Pages 1595-1623,
   ISSN 1319-1578,
   https://doi.org/10.1016/j.jksuci.2021.11.019.

2. Norah N. Alajlan and Dina M. Ibrahim
   TinyML: Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications
   https://doi.org/10.3390/mi13060851

3. Autres : des blogs technologiques, des forums de discussion sur l'IoT et le TinyML…