CESI
ÉCOLE D'INGÉNIEURS

CESI LINEACT

RAPPORT DE STAGE

Mise en place d'un système de Federated Learning

à base de Raspberry Pi

THOMAS Paul

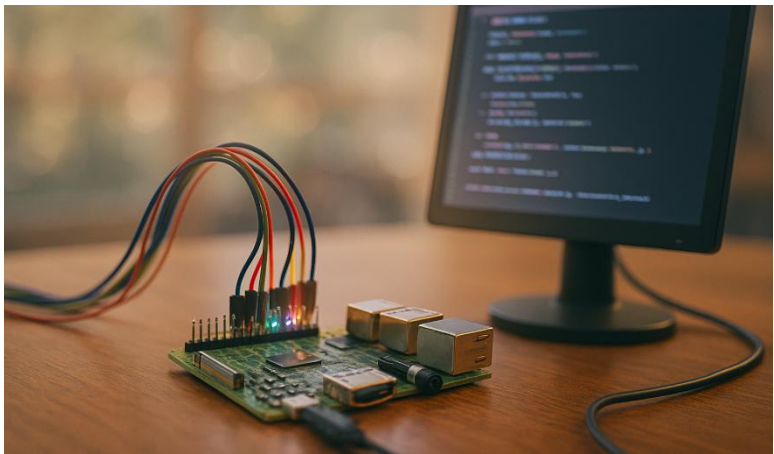
Avril 2025 à juin 2025

Tuteurs du stage : DOLLINGER Jean-François, BRAHMIA Amine et BAAHMED Ahmed

Enseignant référent : ESSAID-FARHAT Amira

Formation : CESI – Deuxième année en cycle préparatoire intégré

Organisme d'accueil : CESI LINEACT - 6 rue de la Faisanderie, 67380 Lingolsheim



Fiche de confidentialité - *Non-disclosure agreement*

Ce document doit être complété pour tout rapport ou mémoire diffusé à CESI École d'Ingénieurs et contenant des informations sur l'entreprise d'accueil.
Ce document est établi en trois exemplaires : un sera conservé par CESI École d'Ingénieur, un autre par l'entreprise, le troisième devra impérativement être intégré au rapport par l'élève.

This page must be included in any document prepared for evaluation purposes at CESI Graduate School of Engineering and that contains information about a host company.
It is produced in three copies: one for the school's records, one for the company, and one which must be included in the student's report.

TITRE DU DOCUMENT EN FRANCAIS	**Rapport de stage – Mise en place d'une architecture distribuée**
Title of the document in English	Internship report – Implementation of a distributed architecture
NOM DE L'ETUDIANT *Student's Name*	THOMAS Paul
PROGRAMME *Program*	CPI A2 – Info
ENTREPRISE D'ACCUEIL *Host company*	CESI LINEACT
TUTEUR(S) DE STAGE *In-company supervisor*	DOLLINGER Jean-François, BRAHMIA Amine et BAAHMED Ahmed

☒ **DIFFUSION LIBRE** ***Full disclosure***	Les rapports / mémoires sont conservés en archives et ils peuvent être librement consultés. Ils peuvent être utilisés par les destinataires, les études peuvent faire l'objet de publication … *The documents are kept in the school's archives and can be consulted freely. They can be used by anyone who receives them and the studies they contain can be published…*
☐ **DIFFUSION RESTREINTE** ***Limited disclosure***	Les rapports / mémoires **sont restitués à l'entreprise** à l'issue de la soutenance. Aucune reproduction n'est autorisée. La responsabilité de cette opération est confiée au stagiaire. Les informations communiquées oralement aux membres du jury lors de la soutenance font également l'objet d'une obligation de confidentialité. A ce titre, les membres du jury s'engagent à ne pas divulguer ni utiliser, toute information qu'ils seraient amenés à connaître pendant une durée indéterminée. *The documents **will be returned to the company** at the end of the oral presentation. No copies are allowed. Retrieval of the document is the sole responsibility of the student. The information communicated orally to the members of the assessment board during the oral presentation is also subject to confidentiality. The members of the board therefore commit to not divulging or using any information that may come to their knowledge for an indefinite period of time.*

Dans le cadre de la politique de lutte contre le plagiat, les rapports / mémoires seront susceptibles d'être analysés pour en vérifier les sources et ceci quel que soit le mode de diffusion prévu ci-dessus.

As part of our anti-plagiarism policy, it is likely that reports will be analysed to verify the sources, regardless of the level of confidentiality specified above.

Date:19/06/2025	Date: 05/06/2025	Date:
L'entreprise / The company	L'étudiant/ The student	L'école / The school

Remerciements

Je voudrais avant tout remercier chaleureusement toute l'équipe de CESI LINEACT pour m'avoir accueilli au sein de leur structure durant ces derniers mois.

Un merci tout particulier à Jean-François, Amine et Ahmed qui m'ont encadré tout au long de ce stage. Leur disponibilité, leurs conseils et la confiance qu'ils m'ont accordée ont été déterminants pour la réussite de ce projet. Ils ont su me guider quand j'en avais besoin tout en me laissant l'autonomie nécessaire pour apprendre.

Je tiens aussi à remercier tous les chercheurs, doctorants et stagiaires présents pour leur bienveillance qui a vraiment contribué au bon déroulement de cette expérience.

Enfin, mes remerciements vont également à Amira, mon enseignante pédagogique, qui m'a aidé à décrocher ce stage et qui a toujours été disponible pour répondre à mes questions pendant ces derniers mois.

Résumé

Français

Ce rapport présente mon projet réalisé au laboratoire de recherche CESI LINEACT, sur le campus de Strasbourg, et qui visait à mettre en place une infrastructure distribuée[1] dédiée à l'apprentissage fédéré (Federated Learning ou FL)[2]

Mon objectif était de concevoir un système d'apprentissage automatique distribué, basé sur un réseau de Raspberry Pi[3] connectés à un serveur central, afin d'améliorer les performances d'entraînement de modèles[4] tout en conciliant vitesse de convergence et charge de calcul.

Ce rapport retrace les différentes étapes de ce projet, de la recherche et documentation à l'optimisation des performances, en passant par l'implémentation du protocole de Federated Learning, la mise en place d'un système de sélection des clients, l'installation du matériel et la configuration du réseau.

English

This report presents my project at the CESI LINEACT research laboratory on the Strasbourg campus, aimed at setting up a distributed infrastructure dedicated to Federated Learning (FL).

My objective was to design a distributed machine learning system, based on a network of Raspberry Pi connected to a central server, in order to improve model training performance while reconciling convergence speed and computing load.

This report outlines the various stages of the project, from research and documentation to performance optimization, including implementation of the Federated Learning protocol, setting up the client selection, hardware installation and network configuration.

Table des matières

I. Introduction

Dans notre monde, l'intelligence artificielle (IA)[5] est partout autour de nous : dans nos téléphones qui reconnaissent notre voix, nos voitures qui nous aident à nous garer, les applications qui nous recommandent des films, les outils médicaux qui aident les médecins à faire des diagnostics et bien plus encore ! Cette technologie s'est imposée dans de nombreux aspects de la vie courante, en facilitant et en automatisant de multiples tâches. Pour que l'IA soit performante, elle doit se nourrir constamment de données. Plus les données à disposition de l'intelligence artificielle sont nombreuses et variées, plus sa capacité à fournir des résultats pertinents s'améliore.

Cependant, malgré ses avancées, l'entraînement de ces intelligences artificielles reste une tâche complexe et loin d'être idéale. Aujourd'hui, les méthodes les plus couramment utilisées reposent sur la collecte d'un large volume de données, issues de diverses sources telles que les téléphones, les capteurs, les objets connectés ou les plateformes en ligne. Ces données sont ensuite transférées vers des centres de calcul centralisés pour y être analysées et traitées. Bien que cette approche permette un apprentissage efficace, elle présente plusieurs limites, notamment en ce qui concerne l'impact environnemental, la protection des données ainsi que la scalabilité et les performances. La centralisation implique une charge de calcul importante sur les serveurs, une forte consommation de ressources, et des délais de traitement qui peuvent s'avérer non négligeables. Dès lors, se pose la question de la recherche de compromis entre la vitesse de convergence des modèles d'apprentissage et la charge computationnelle induite, notamment dans des contextes où les ressources sont limitées ou réparties de manière hétérogène.

C'est là qu'intervient l'apprentissage fédéré ou *Federated Learning* en anglais. Cette approche représente une vraie révolution dans la façon de concevoir l'intelligence artificielle. Elle change complètement le fonctionnement centralisé classique. Concrètement, l'apprentissage fédéré consiste à entraîner les modèles directement sur les dispositifs des utilisateurs (téléphones, ordinateurs, objets connectés, véhicules…), plutôt que de centraliser les données sur un serveur distant. Ainsi, les données restent stockées localement, tandis que seuls les paramètres ou les mises à jour du modèle, résultant des calculs effectués en local, sont transmis au serveur central.

Cette méthode apporte beaucoup d'avantages. D'abord, elle protège notre vie privée car nos données personnelles ne quittent jamais nos appareils. Ensuite, elle permet d'exploiter efficacement la puissance de calcul déjà disponible sur nos téléphones et ordinateurs. Plutôt que d'investir dans de nouveaux centres de données à forte consommation énergétique, cette approche tire parti des ressources informatiques existantes. Cette approche s'inscrit dans une logique décentralisée : pourquoi centraliser tous les calculs dans quelques endroits quand on a des milliards d'appareils connectés capables de traiter des données ? Grâce à l'apprentissage fédéré, il est possible de faire progresser l'IA tout en respectant à la fois la confidentialité des utilisateurs et les enjeux environnementaux.

C'est dans ce contexte passionnant que j'ai réalisé mon stage au laboratoire CESI LINEACT, campus de Strasbourg. L'idée de contribuer à cette technologie d'avenir me motivait énormément. Ma mission, à la fois bien définie et ambitieuse, consistait à créer et implémenter de A à Z une infrastructure d'apprentissage fédérée en utilisant plusieurs Raspberry Pi. Ces petits ordinateurs de la taille d'une carte de crédit allaient simuler des clients[6] intelligents, comme s'il

s'agissait de téléphones ou d'objets connectés en conditions réelles. Mais comment faire coopérer ces petits appareils, pas très puissants individuellement mais nombreux, pour qu'ils travaillent ensemble efficacement ? Coordonner plusieurs appareils qui communiquent entre eux, gérer les pannes, optimiser les performances… Tout cela représentait des défis techniques que je n'avais jamais abordés. Mais c'est justement ce qui rendait le projet de mon stage si stimulant.

Ce rapport retrace les étapes du déploiement d'un algorithme de sélection de client sur une infrastructure FEEL (*efficacy of a Federated Edge Learning*) à base de Raspberry Pi. Vous allez découvrir premièrement le laboratoire CESI LINEACT, les missions qui ont été confiées, ainsi que le planning établi. Par la suite, nous aborderons certaines notions fondamentales que j'ai découvertes dans mes premières recherches. Vous allez découvrir concrètement comment j'ai mis en place le système de Federated Learning avec ses premiers tests concluants, en passant par différentes grandes étapes et missions chronologiques. Vous découvrirez notamment les choix techniques que j'ai dû faire, les optimisations que j'ai mises en œuvre pour tirer le meilleur parti de mes Raspberry Pi, et aussi les difficultés que j'ai rencontrées en chemin. Enfin, les résultats de tout le travail que j'ai réalisé seront présentés à travers des démonstrations concrètes du système en fonctionnement, des résultats d'expérimentations que j'ai pu analyser, et un comparatif entre les différents paramètres de simulation testés, accompagnés d'une conclusion.

Mon objectif dans ce rapport est de vous illustrer concrètement comment des petits appareils peuvent, en travaillant ensemble de manière intelligente, créer un système d'intelligence artificielle à la fois performant et respectueux de l'environnement. Il s'agit peut-être d'un aperçu des réseaux intelligents distribués que nous verrons apparaître massivement dans les années à venir, quand chaque objet connecté participera à l'intelligence collective tout en préservant nos données personnelles.

II. Présentation du laboratoire CESI LINEACT

2.1. Le laboratoire et ses axes de recherche

Pendant mon stage, j'ai eu l'occasion de travailler au CESI LINEACT (Laboratoire d'Innovation Numérique pour les Entreprises et les Apprentissages au service de la Compétitivité des Territoires) sur le campus de Strasbourg. C'est un laboratoire de recherche appliquée rattaché à l'école d'ingénieurs CESI. Créé en 2015, il a pour but de faire le pont entre la recherche académique et les besoins des entreprises. Il a été reconnu Unité de Recherche en 2018.

Ce qui distingue CESI LINEACT est l'approche pluridisciplinaire mobilisée dans les projets de recherche associant sciences du numérique, sciences de l'ingénieur et sciences humaines et sociales. En effet, les innovations technologiques ne peuvent être pleinement efficaces que si elles répondent aux besoins réels des utilisateurs, en intégrant leurs pratiques, leurs attentes et leurs contraintes. C'est exactement ce que j'ai pu observer pendant mon stage : l'apprentissage fédéré, c'est autant une question technique qu'une question d'usage.

Le laboratoire LINEACT est présent partout en France avec une grande envergure. Voici quelques chiffres sur l'effectif du laboratoire :

10	110	2	16	74	3
Directeurs de recherche	Enseignants chercheurs	Cadres administratifs et financiers	Ingénieurs de recherche	Doctorants	Plateformes de recherche

Au total, c'est plus de 200 personnes qui travaillent ensemble partout en France sur quatre grands domaines d'avenir :

- L'Industrie 5.0 : l'usine de demain, plus humaine et plus durable
- La construction 4.0 et la ville durable : comment construire et habiter autrement
- Les formations du futur : réinventer la façon d'apprendre et d'enseigner
- Les services numériques : développer des outils digitaux vraiment utiles

CESI LINEACT dispose également de trois plateformes technologiques à Rouen et Nanterre. Deux sont dédiées à l'usine du futur et une au bâtiment du futur.

Ce qui est également notable ici, c'est la proximité avec les besoins des industries et des entreprises. La recherche qui se fait ici n'est pas déconnectée des réalités économiques. Les projets naissent souvent d'un besoin exprimé par une entreprise, et les résultats sont utilisés directement par celle-ci. C'est également une chose qui m'a beaucoup plu pendant mon stage. Travailler sur l'apprentissage fédéré, c'était très motivant sur le plan technique, mais savoir que mon travail pourrait aider pour un projet concret donnait encore plus de sens à mes efforts.

Mon sujet de stage fait partie des projets portés par l'unité de recherche de CESI LINEACT. Il répond à des enjeux actuels liés à l'optimisation des performances en intelligence artificielle distribuée. J'ai eu la chance d'intégrer l'équipe ION (« Ingénierie et Outils Numériques »), l'une des deux grandes équipes de recherche du laboratoire. J'ai été encadré par trois chercheurs spécialisés en Edge-Cloud, systèmes distribués et IoT : Jean-François Dollinger, Amine Brahmia et Ahmed Baahmed. Leur accompagnement ainsi que leurs expertises m'ont grandement aidé dans la réalisation et la progression de mon travail tout au long de ce stage.

2.2. Missions confiées

Avant d'entamer les développements techniques, il était essentiel de clarifier les attendus du stage. Dès les premiers échanges avec mes tuteurs, nous avons défini ensemble les objectifs principaux, afin d'orienter le travail de manière structurée et cohérente avec les enjeux du laboratoire. Les objectifs fixés étaient les suivants :

- Mettre en place une infrastructure distribuée combinant plusieurs Raspberry Pi et un serveur central.
- Implémenter le système de Federated Learning en s'appuyant sur une architecture distribuée.
- Déployer les algorithmes existants de sélection de clients FL afin d'optimiser les performances calculatoires.

A partir de ces objectifs, le travail a été découpé en cinq grandes missions successives :

1. Recherche documentaire et prise en main des concepts du Federated Learning.
2. Installation du matériel, configuration du réseau, adressage IP et configuration des Raspberry Pi.
3. Développement et implémentation du système de Federated Learning, création du protocole de communication et algorithme de sélection de clients.
4. Optimisation des performances de calcul, du réseau et identification des goulots d'étranglement du processus entier afin de l'améliorer.
5. Validations, scénarios de tests pour valider l'efficacité de l'implémentation, interface web et documentation du projet.

2.3. Planning prévisionnel

Pour structurer l'ensemble des missions identifiées, j'ai établi un diagramme de Gantt dès le début du stage. Cette planification a permis de répartir les différentes phases de travail sur les douze semaines disponibles, en tenant compte des dépendances entre les tâches et des étapes clés du projet.

L'objectif de ce planning était de garantir une progression cohérente, en évitant les blocages liés à un manque d'anticipation ou à une surcharge de travail. Il a également servi de support pour suivre l'avancement du projet de manière régulière.

Même si le cadre initial a été respecté dans les grandes lignes, plusieurs ajustements ont été nécessaires en fonction des contraintes techniques rencontrées, des délais d'installation ou de configuration.

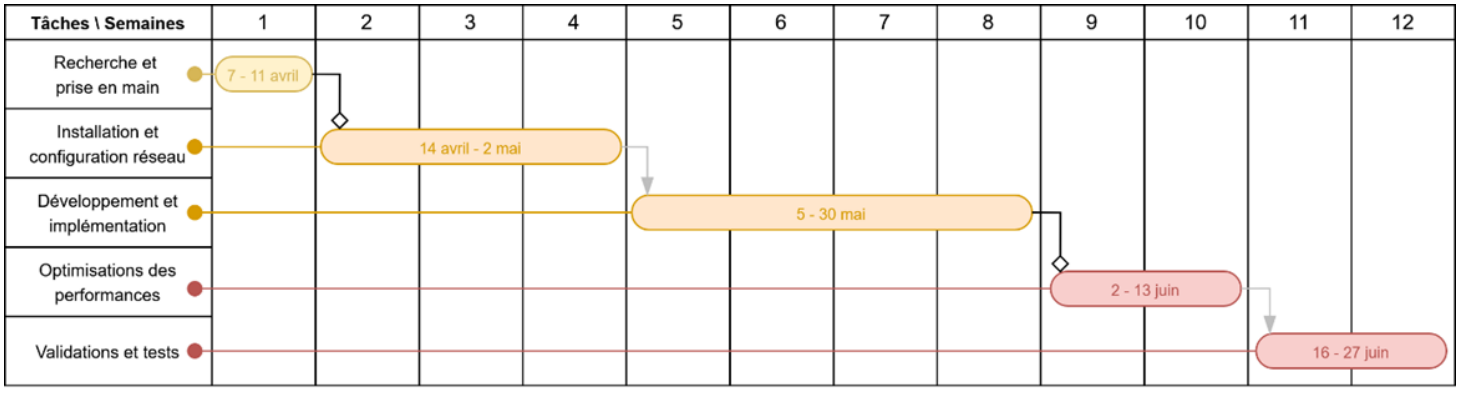


Figure 1 : Planning de Gantt prévisionnel du projet

III. Concepts clés de l'intelligence artificielle

Avant de poursuivre ce rapport, il est essentiel d'introduire certaines notions fondamentales qui seront régulièrement évoquées par la suite.

Tout d'abord, l'intelligence artificielle. Elle désigne un ensemble de systèmes informatiques capables d'accomplir des tâches traditionnellement associées à l'intelligence humaine, telles que la reconnaissance vocale, la traduction automatique, ou encore la recommandation personnalisée de contenus sur Internet.

L'un des types d'intelligence artificielle les plus courants aujourd'hui, est appelé apprentissage automatique[7] ou *Machine Learning (ML)* en anglais. En général, un algorithme classique est prédéterminé et n'évolue pas au cours du temps. L'intérêt du Machine Learning est de caractériser avec souplesse une fonction complexe, difficilement modélisable à l'aide d'un ensemble de règles. Il va apprendre à accomplir une tâche sans qu'on lui programme explicitement chaque étape. Pour cela, on lui donne des données, beaucoup de données ! Prenons un exemple concret : on souhaite que notre ordinateur reconnaisse des photos de chiens et de chats. Au lieu de lui expliquer « un chien a de grandes oreilles, un museau, … », nous lui montrons simplement des milliers d'images déjà étiquetées « ça, c'est un chien », « ça, c'est un chat » … Grâce à tous ces exemples, l'ordinateur va progressivement ajuster ses paramètres pour faire des prédictions de plus en plus précises.

L'apprentissage profond[8] ou Deep Learning en anglais, constitue une sous-catégorie du Machine Learning qui utilise des réseaux de neurones artificiels[9] avec plusieurs couches cachées, d'où le terme « deep » (profond). C'est cette technologie qui permet aux IA modernes de comprendre des données vraiment complexes, comme des images haute résolution ou du langage naturel.

Et c'est exactement sur cette catégorie que j'ai travaillé pendant tout mon stage !

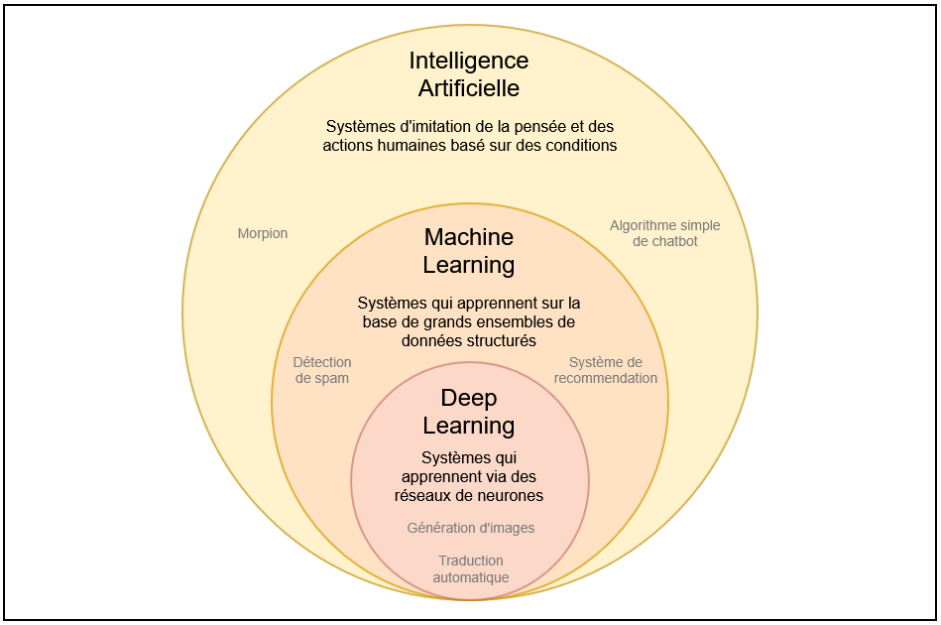


Figure 2 : Schéma comparatif des concepts d'IA

Le Deep Learning s'appuie donc sur un réseau de neurones[10], une notion clé. C'est un modèle mathématique inspiré du cerveau humain. Il sert à faire des prédictions à partir de données, et est composé de plusieurs couches de petits calculateurs, appelés neurones artificiels. Chaque neurone réalise un calcul : il combine les entrées en les multipliant par des poids[11] qui indiquent l'importance de chaque information, et renvoie un résultat. Des biais ou fonctions d'activation sont d'autres paramètres fréquemment ajoutés au calcul en fonction du modèle d'IA.

$$sortie = (entrée * poids) + paramètres\ supplémentaires\ (ReLU, \dots)$$

Il existe deux grands types d'apprentissage : supervisé, où les données sont étiquetées (par exemple des images avec leur catégorie), et non supervisé, où le modèle doit découvrir seul des structures ou regroupements dans les données.

Parmi les réseaux de neurones, les CNN (Convolutional Neural Networks) sont particulièrement adaptés aux images. Ils utilisaient des couches appelées convolutions qui détectent automatiquement des motifs locaux (bords, formes, …) avant de les combiner pour reconnaître des objets entiers.

Ce réseau de neurones apprend progressivement à effectuer des prédictions pertinentes grâce à une phase d'entraînement. Reprenons l'exemple précédent. Le réseau de neurones va donc recevoir une base de données d'images déjà étiquetées, puis il va faire plusieurs fois :

- Une prédiction : « Je pense que cette image est un chat. »
- Il compare avec la bonne réponse : « Ah non, c'était un chien. »
- Il ajuste les poids pour se rapprocher de la bonne réponse la prochaine fois.

Ce processus s'appelle la rétropropagation, car l'erreur remonte dans tout le réseau pour corriger les poids de tous les neurones. Voici un schéma permettant de visualiser un réseau de neurones, utilisé en Deep Learning :

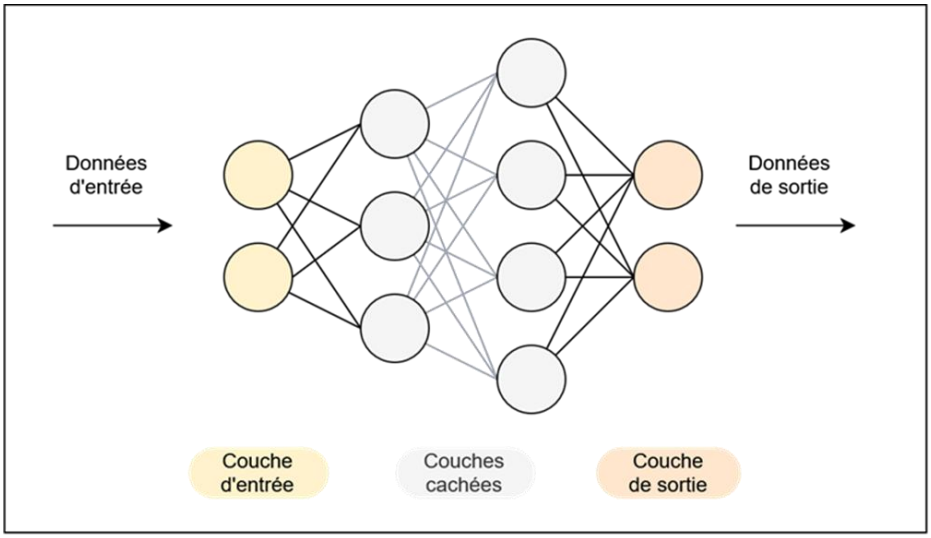


Figure 3 : Schéma d'un réseau de neurones

Avant de me lancer dans le développement et les aspects techniques de mon stage, je voulais utiliser ma première semaine pour bien comprendre le fonctionnement d'un système de Federated Learning, et je ne le regrette pas !

J'ai donc commencé, dès ma première semaine de stage, ma recherche documentaire approfondie sur l'apprentissage fédéré, mais aussi sur des concepts plus larges, comme le Deep Learning ou encore l'intelligence artificielle. Parce que pour bien maîtriser l'apprentissage fédéré, il faut d'abord comprendre comment fonctionne l'apprentissage centralisé[12] ou Centralized Learning en anglais.

IV. Limites de l'apprentissage centralisé

L'apprentissage centralisé, encore majoritairement utilisé aujourd'hui, soulève de nombreuses interrogations. Techniquement, cette approche centralise tous les traitements dans un serveur unique, en collectant les données depuis des appareils distants avant de les traiter. Toutes les données des appareils sont envoyées vers un serveur central, puis celui-ci va entraîner le modèle d'intelligence artificielle à partir des données reçues.

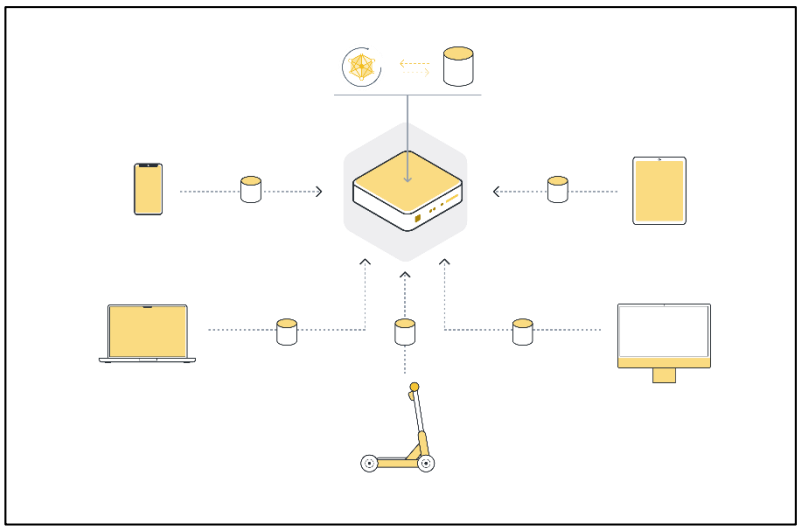


Figure 4 : Schéma d'un apprentissage centralisé (Centralized Learning) ***Source :*** (Flower AI)

Cette approche soulève d'importants problèmes, tant sur l'aspect de la protection des données, que sur l'aspect environnemental. L'entraînement d'un réseau de neurones comportant des centaines de milliards de paramètres comme ceux utilisés dans les modèles de pointe actuels, peut s'étaler sur plusieurs mois et représenter un coût financier pouvant atteindre plusieurs dizaines de millions d'euros. (BDM, 2024) De nombreux travaux scientifiques ont déjà démontré que leur impact environnemental est très important, sans parler de l'énorme quantité de données personnelles récoltées pour entraîner ces modèles. (ISE, 2024)

V. Principes et enjeux de l'apprentissage fédéré

Le Federated Learning est une approche visant à résoudre cette double problématique. Dans ce cadre, Jean-François, Amine et Ahmed ont développé une implémentation spécifique de cette méthode, sur laquelle j'ai eu l'opportunité de travailler pendant mon stage. Ce principe change les règles de l'apprentissage classique : au lieu d'envoyer les données vers un serveur central, c'est désormais le serveur qui transmet un modèle d'intelligence artificielle aux appareils distants. Chacun de ces appareils, appelés clients, entraîne alors localement le modèle sur ses propres données. Une fois cette phase terminée, chaque client renvoie uniquement les paramètres mis à jour du modèle au serveur. Ce dernier procède ensuite à une agrégation[13] de l'ensemble des modèles reçus, afin de construire un modèle global plus performant. Si le serveur juge ce modèle suffisamment efficace, l'apprentissage fédéré s'achève ; dans le cas contraire, un nouveau cycle commence. Ce processus se répète jusqu'à ce que le modèle atteigne un niveau de performance jugé satisfaisant.

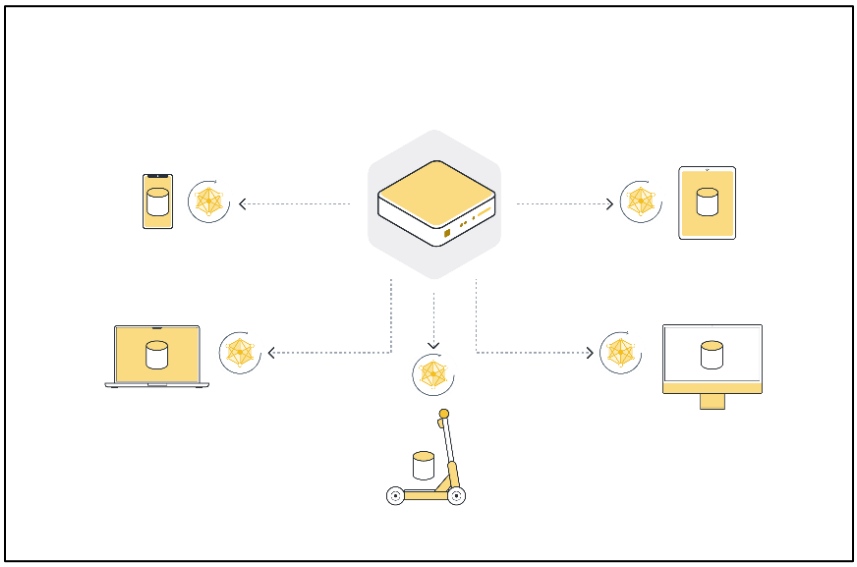


Figure 5 : Schéma d'un apprentissage fédéré (Federated Learning) ***Source :*** *(Flower AI)*

Grâce à cette approche, nous n'avons plus besoin d'un serveur avec d'énormes performances, car ce dernier ne fait plus aucun apprentissage intensif. Le travail est réparti sur de nombreux clients qui sont, dans tous les cas, déjà sollicités dans les usages quotidiens des utilisateurs, comme c'est le cas d'une station météo opérant en continu, par exemple. Cela permet de résoudre les problématiques de confidentialité des données et de réduction de l'empreinte carbone.

VI. Conception et mise en œuvre de l'architecture distribuée

Après mes recherches documentaires, j'ai commencé mon travail dès la deuxième semaine sur Python, le langage de programmation le plus populaire pour faire de l'intelligence artificielle. J'ai réalisé un premier programme avec la librairie Tensorflow, dans le but de reconnaitre et de prédire des chiffres manuscrits. Grâce aux conseils de mes tuteurs de stage, j'ai su implémenter un réseau de neurones conçu en amont. Ce dernier a été entraîné sur la base de données publique MNIST, qui regroupe plus de 60 000 images d'apprentissage. Ce sont des images en noir et blanc, normalisées, centrées de 28 pixels de côté.

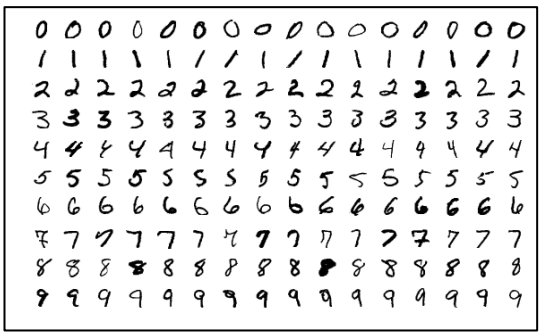


Figure 6 : Aperçu de la base de donnée MNIST

Source : (Wikipedia, 2017)

En changeant la structure de mon réseau de neurones, nous pouvons voir l'impact sur le temps d'apprentissage, sur la précision de reconnaissance du modèle et nous pouvons saisir beaucoup d'autres aspects importants à comprendre. Cela m'a permis d'assimiler les concepts de base des réseaux de neurones et de les utiliser avec un cas concret.

6.1. Communication du système

Une fois le réseau de neurones implémenté et entraîné avec la base de données MNIST, l'objectif était d'implémenter ce dernier sur un réseau, afin de décentraliser l'apprentissage sur plusieurs appareils. L'un des aspects les plus critiques en Federated Learning concerne la communication entre les appareils. En effet, un modèle d'intelligence artificielle peut être assez volumineux et doit être transféré en aller-retour pour chaque appareil connecté à notre réseau, et ce pour chaque cycle du Federated Learning.

TCP	→ Protocole de transport bas niveau, orienté connexion. → Garantit la livraison et l'intégrité des paquets (réémission si besoin). → Peu adapté pour transmettre directement des données structurées.
UDP	→ Protocole de transport bas niveau, non orienté connexion. → Très rapide avec une faible latence. → Aucune garantie de livraison, difficile à exploiter pour des échanges fiables ou structurés.
HTTP(S)	→ Protocole de haut niveau basé sur TCP, orienté requête/réponse. → Simple à utiliser, largement supporté, idéal pour des communications ponctuelles. → Unidirectionnel, latence moyenne, moins optimisé pour le temps réel.
WebSocket	→ Protocole basé sur TCP, permettant une communication bidirectionnelle persistante. → Faible latence, fiable, et bien adapté pour des échanges en temps réel. → Mise en place plus simple que TCP brut, mais nécessite un support côté client et serveur.
MQTT	→ Protocole léger de type publish/subscribe, reposant sur TCP via un broker. → Très utilisé dans l'IoT, faible latence et bonne fiabilité sur des connexions peu stables. → Simple à configurer, passe par le broker pour distribuer les messages.
gRPC	→ Protocole plus haut niveau (couche application) basé sur HTTP/2. → Latence extrêmement faible, conçu pour des échanges structurés et performants. → Plus complexe à mettre en place, nécessite une définition stricte des messages.

Mon objectif était d'utiliser une technologie adaptée au temps réel, suffisamment légère pour de petites machines, et surtout avec une faible latence. J'ai directement éliminé UDP ainsi que tous les protocoles basés sur ce dernier, car la transmission de tous les paquets n'y est pas garantie. Il est inconcevable, dans le cadre du FL, de transmettre un modèle d'IA incomplet.

Je me suis ensuite tourné vers le protocole TCP, un protocole de transport bas niveau fiable et largement utilisé. J'ai donc implémenté un petit programme, côté serveur et côté client, pour transmettre des données avec la librairie sockets de Python. Cependant, j'ai rapidement réalisé la complexité que cela impliquait. Avec un protocole bas niveau comme celui-ci, il faut savoir tout gérer soi-même : fragmenter les données initiales en petits paquets, créer des algorithmes de vérification pour une potentielle retransmission, gérer les messages réseau pour les déconnexions silencieuses…

Rapidement, je me suis donc plutôt orienté vers des protocoles plus simples à implémenter et tournés vers la communication en temps réel. Le protocole HTTP me paraissait très efficace, mais en essayant de l'appliquer, il s'est révélé trop coûteux en surcharge réseau, notamment à cause des entêtes pour des petits appareils et peu adapté à des échanges fréquents en temps réel.

Grâce à mes tuteurs de stage, j'ai découvert MQTT, un protocole de messagerie léger très utilisé en IoT pour faire communiquer plusieurs appareils entre eux. Ce protocole fonctionne avec un broker (traitement central) qui reçoit et redistribue les messages. Grâce à cela, le serveur n'a plus besoin d'établir une connexion avec tous les clients connectés : il récupère simplement les messages déposés par les clients dans le broker. Le seul inconvénient réside dans le sens inverse de la communication, lorsque le serveur souhaite envoyer des messages aux clients. En effet, lorsqu'un message est publié sur un topic partagé, tous les clients abonnés le reçoivent, le traitent et sont donc impactés sur leurs performances.

C'est pourquoi, après réflexion, j'ai décidé de combiner MQTT avec WebSocket, un autre protocole de communication adapté au temps réel, mais beaucoup plus approprié pour de la communication de pair à pair. Il présente une latence encore plus faible que celle de MQTT et reste relativement simple à configurer. En utilisant cette approche hybride, j'ai combiné les avantages des deux protocoles pour obtenir les meilleures performances. La figure 7 ci-dessous illustre l'infrastructure et les méthodes de communication choisies.

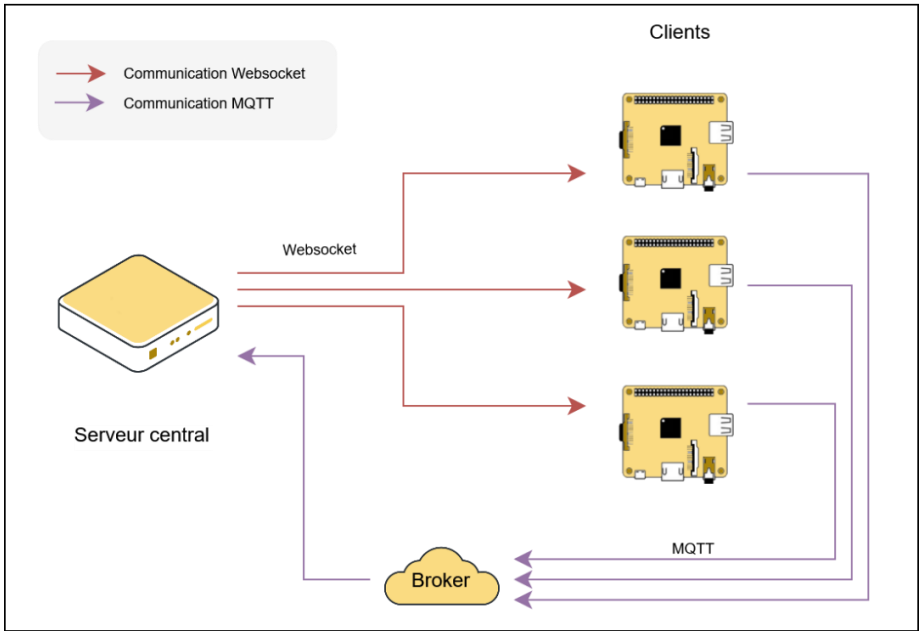


Figure 7 : Schéma des communications sur l'infrastructure

Cette architecture de communication que j'ai conçue permettra à mes appareils de communiquer facilement au serveur tout en étant capables de recevoir des messages de ce dernier. Et tout cela, en impactant très peu leur performance. Cette solution est particulièrement bien adaptée à notre système de Federated Learning, tout en restant suffisamment flexible pour être étendue à un plus grand nombre d'appareils à l'avenir.

6.2. Configuration du réseau et du matériel

Une fois le système de communication mis en place, j'ai souhaité le tester en situation réelle avec le matériel mis à disposition par mes tuteurs. Mes tuteurs m'ont confié 5 Raspberry Pi avec des versions différentes ainsi qu'un routeur. J'ai également apporté mon Raspberry Pi personnel pour avoir plus de clients pour mes tests. J'ai donc d'abord configuré le routeur avec un adressage IP statique pour éviter tout problème de communication entre mes appareils. Dans le cadre de cette simulation, j'ai utilisé mon ordinateur personnel en tant que serveur central, ce qui fonctionnait parfaitement puisque le Federated Learning ne nécessite pas de serveur particulièrement performant.

Ensuite, j'ai procédé à la configuration des Raspberry Pi, en commençant par le choix du système d'exploitation. Je recherchais un OS suffisamment léger pour fonctionner sur tous les appareils (dotés de 1 à 4 Go de RAM), mais assez robuste pour faire tourner de gros réseaux de neurones. J'ai hésité entre plusieurs distributions Linux populaires : Ubuntu, Fedora, Debian, CentOS… Le premier choix me paraissait être idéal avec sa version Ubuntu Server (version allégée en ligne de commande uniquement). Ce système d'exploitation offrait un ensemble de commandes très utiles, un environnement Python prêt à l'emploi, et surtout une communauté active garantissant la stabilité du système avec des mises à jour régulières.

Cependant, lors de l'exécution des apprentissages d'IA sur les clients, j'ai constaté que certains Raspberry Pi rencontraient de nombreuses erreurs de mémoire insuffisante. Malgré mes efforts pour optimiser au maximum mon programme, le simple import de la bibliothèque Python dédiée à l'IA posait déjà des problèmes. À ce moment-là, j'ai eu quelques sueurs froides en réalisant la difficulté que représente l'entraînement d'un modèle d'intelligence artificielle complexe sur de si petits appareils.

J'ai alors testé Arch Linux, une autre distribution reconnue pour ses excellentes performances, mais cela n'y changeait rien. Je me suis donc tourné vers les forums Raspberry pour trouver une solution à cette consommation de RAM élevée. C'est là que j'ai découvert les avantages de Raspberry Pi OS Lite, un système d'exploitation moins connu mais qui présente de nombreux atouts. Il est spécialement optimisé et adapté aux Raspberry Pi, et offre en plus des options de paramétrage simples pour configurer les appareils.

Après l'avoir installé et testé à nouveau avec mon programme, tout fonctionnait ! Il n'y avait plus d'erreur de mémoire, du moins pour l'instant. Cette étape cruciale de mon projet étant terminée, j'ai réalisé un tableau récapitulatif des informations importantes concernant la configuration matérielle et réseau, afin de m'y retrouver par la suite :

IP	Nom	Appareil	Système d'exploitation	Hostname	Ram	Adresse MAC
192.168.1.1	Routeur	Routeur Cisco Linksys	/	/	/	/
192.168.1.100	Serveur	Mon ordinateur	x	/	x	x
192.168.1.101	Client 1	Raspberry Pi Model 2	Raspberry Pi OS Lite (32 bits)	client01.local	1 Go LPDDR2	a4:98:lc:pf:03:42
192.168.1.102	Client 2	Raspberry Pi model 4B+	Raspberry Pi OS Lite (64 bits)	client02.local	4 Go LPDDR4	dc:a6:32:4c:c1:1d
192.168.1.103	Client 3	Raspberry Pi model 4B+	Raspberry Pi OS Lite (64 bits)	client03.local	4 Go LPDDR4	dc:a6:32:4c:fd:98
192.168.1.104	Client 4	Raspberry Pi model 3B	Raspberry Pi OS Lite (64 bits)	client04.local	1 Go LPDDR2	b8:27:eb:86:f1:6f
192.168.1.105	Client 5	Raspberry Pi model 3B	Raspberry Pi OS Lite (64 bits)	client05.local	1 Go LPDDR2	b8:27:eb:e1:57:09
192.168.1.106	Client 6	Raspberry Pi model 4B+	Raspberry Pi OS Lite (64 bits)	client06.local	4 Go LPDDR4	e4:5f:01:ec:89:7f

Légende :

- IP : adresse IP locale attribuée à l'appareil sur le réseau.
- Nom : nom donné à l'appareil pour l'identification (utilisateur ou fonction).
- Appareil : modèle ou type de matériel utilisé.
- Système d'exploitation : système d'exploitation installé sur l'appareil.
- Hostname : nom d'hôte utilisé par l'appareil sur le réseau local.
- RAM : quantité et type de mémoire vive (RAM) de l'appareil.
- Adresse MAC : adresse physique unique de l'interface réseau de l'appareil (Media Access Control).

A ce stade, j'ai réalisé la configuration de l'environnement Python avec l'installation des librairies sur chacun des appareils sans trop de difficulté, car tous les appareils avaient le même système d'exploitation. J'en ai même profité pour faire un script en bash automatisant l'adressage IP sur le réseau, l'installation de l'environnement Python avec ses librairies, la configuration des paramètres, le téléchargement du dataset et beaucoup d'autres étapes. Celui-ci m'a permis de déployer et configurer toutes les Raspberry Pi beaucoup plus facilement, et sera utile si nous devons ajouter des nouveaux appareils dans le futur.

J'ai ensuite développé un programme en Python, côté serveur et côté client, pour pouvoir transférer des messages simples sur les protocoles de communication choisis précédemment. Tout fonctionnait parfaitement, dans les deux sens, que ce soit du serveur vers les clients avec Websocket ou des clients vers le serveur via le broker MQTT.

Figure 8 : Six Raspberry Pi connectés au routeur pendant un apprentissage fédéré

6.3. Protocole de signalisation

Maintenant que les appareils fonctionnaient correctement et qu'ils arrivaient à communiquer sur le réseau local que j'avais créé, il fallait mettre en place le cœur du Federated Learning, la signalisation. Cette dernière représente la logique que va suivre notre système en gérant les différentes étapes de la collaboration entre les clients et le serveur central. Elle détermine quand les clients doivent s'entraîner, quand ils doivent envoyer leurs modèles, et comment le serveur doit réagir par rapport à ces différents signaux pour gérer le modèle global. Cette signalisation fonctionne avec un enchaînement précis de messages et d'événements, qui permettent de synchroniser tous les participants du processus d'apprentissage distribué. Je me suis donc d'abord renseigné pour voir comment les systèmes de Federated Learning sont implémentés, et quelles sont les grandes étapes de cette implémentation.

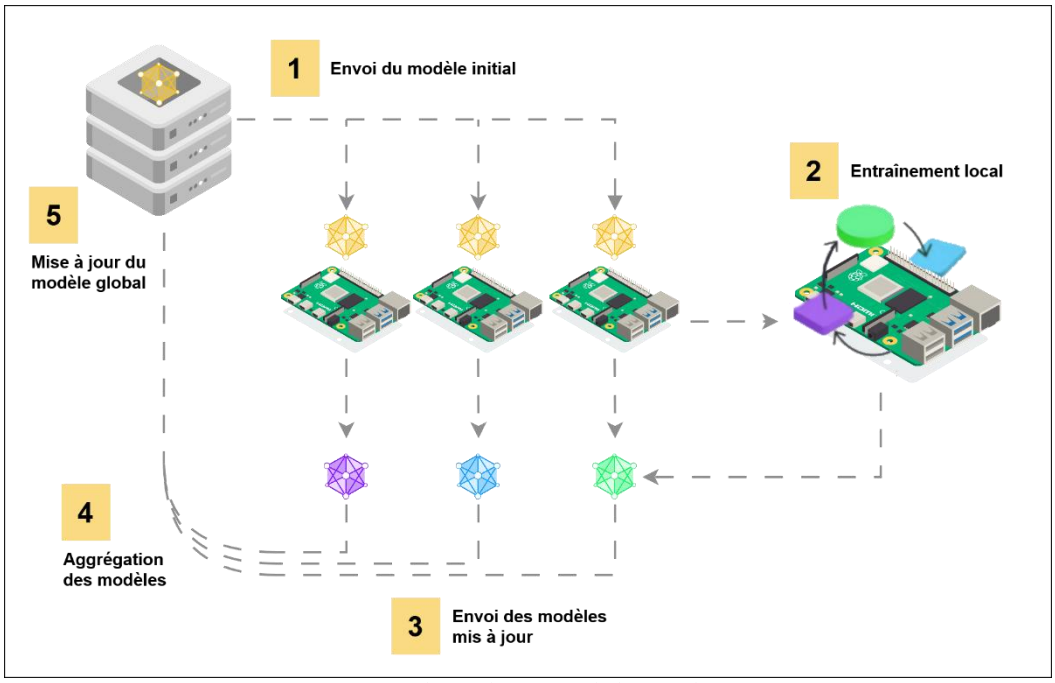


Figure 9 : Schéma des étapes de la signalisation

Voici le fonctionnement en détail :

1. Tout d'abord, un modèle d'IA initialisé dès le début est envoyé à tous les appareils participants.
2. Ensuite, chaque appareil utilise ses propres données locales (comme des images stockées ou des données d'un capteur sur un Raspberry Pi) pour ajuster ce modèle. L'appareil va entraîner le modèle et améliorer ses prédictions à chaque tentative grâce à ses données locales.
3. Une fois l'entraînement local effectué, l'appareil envoie seulement les modifications apportées au modèle (les poids ajustés du modèle). Aucune donnée personnelle ne quitte l'appareil.
4. Le serveur central reçoit les mises à jour de plusieurs appareils et les combine avec un algorithme. Il fusionne les modifications apportées au modèle par chaque appareil, ce qui permet de bénéficier des connaissances ou des points forts de tous les appareils.
5. Le modèle global est mis à jour, pour être ensuite envoyé à tous les appareils. Ce modèle amélioré est maintenant prêt à être utilisé pour une nouvelle phase d'apprentissage.

Tout ce processus se répète de nombreuses fois pour rendre le modèle de plus en plus précis, jusqu'à ce que le serveur considère que le résultat est satisfaisant. Ce résultat peut être atteint lorsque le modèle atteint un certain niveau de précision, lorsque le temps d'apprentissage fixé est écoulé, ou lorsque d'autres critères spécifiques sont atteints. L'ensemble de ces cinq étapes s'appelle un cycle ou un round de Federated Learning.

Une fois que j'avais bien compris le fonctionnement et les grandes étapes d'un processus d'apprentissage décentralisé, j'ai voulu créer un diagramme plus précis, qui me permettrait de créer mon code plus facilement. J'ai opté pour un diagramme de séquence, qui détaille toutes les communications entre les différentes parties du système (serveur, Raspberry Pi, …) et les grandes étapes.

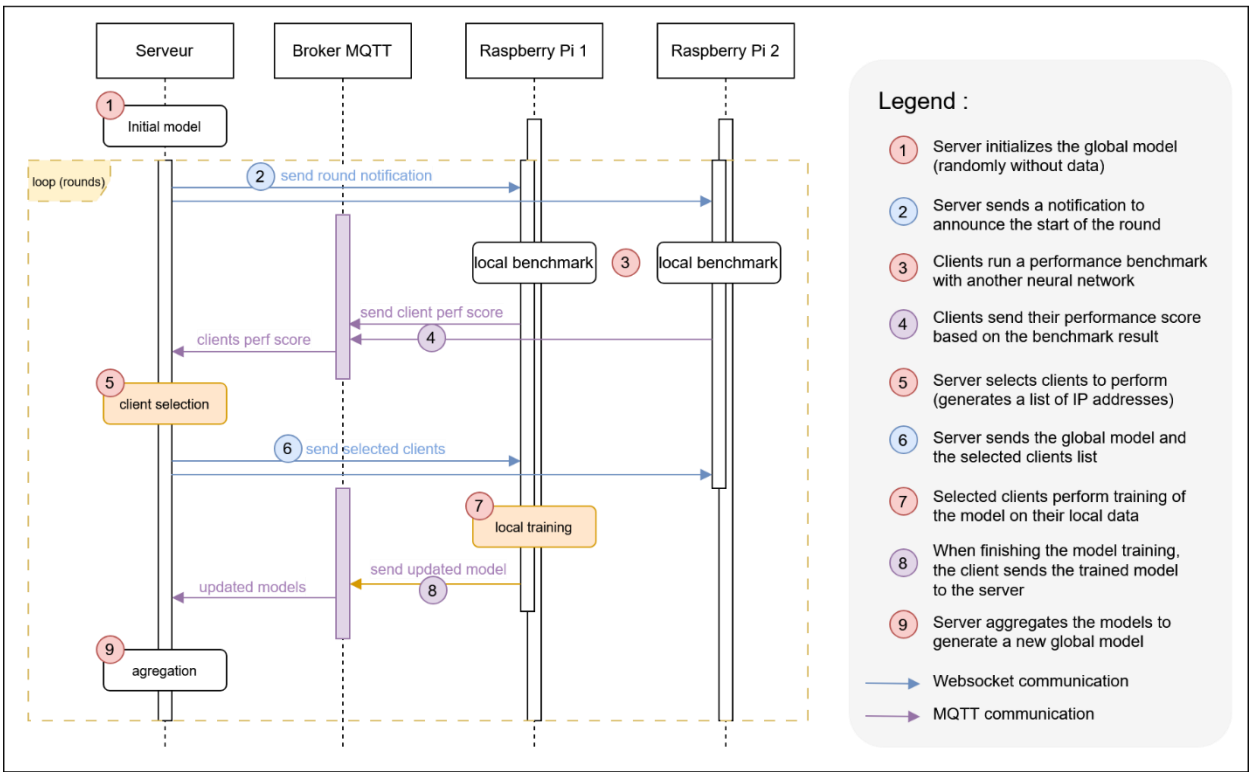


Figure 10 : Diagramme de séquence de la signalétique du FL

En me basant sur les différents programmes que j'avais déjà implémentés en Python, j'ai essayé à ce stade d'implémenter tout le processus FL décrit par le diagramme. Je me suis vite rendu compte, que ce soit côté client ou côté serveur, que le programme devenait très lourd et très dur à débugger. C'est pourquoi j'ai choisi de travailler avec une approche orientée objet, en travaillant avec des fichiers séparés. J'ai créé plusieurs classes et méthodes pour diviser le programme en morceaux plus concis et plus faciles à comprendre. J'ai notamment travaillé avec les classes FederatedClient, FederatedServer, MnistModel, ClientBenchmark et beaucoup de méthodes dans chacune d'elles. Malgré tous ces efforts pour rendre mon code plus lisible, j'ai néanmoins rencontré de nombreuses difficultés pour faire fonctionner le système. J'ai notamment rencontré des problèmes de synchronisation entre les messages, des problèmes de fonction bloquante (attente de la réception entière d'un message) ainsi que des limitations en termes de taille maximale des messages que je pouvais envoyer…

6.4. Sélection de clients

Comme vous l'avez aperçu, une étape supplémentaire est présente dans le diagramme de séquence par rapport à un cycle classique d'apprentissage fédéré, que je n'ai pas encore abordée. En effet, Amine, Ahmed et Jean-François ont réalisé un travail conséquent en développant un algorithme de sélection de clients. Celui-ci permet au serveur de choisir de manière intelligente et dynamique les appareils participant au processus d'apprentissage collaboratif, en s'appuyant sur plusieurs critères clés liés à leurs performances et caractéristiques spécifiques. (Baahmed, Dollinger, Brahmia, & Zghal, 2024)

Le premier critère est la performance matérielle de chaque appareil. Si plusieurs clients peuvent traiter les tâches rapidement, mais qu'un seul est significativement plus lent, alors l'ensemble du système, serveur et clients, doit attendre ce dernier pour passer à la phase suivante. Cela crée un effet de ralentissement global, appelé Straggler Problem[14]. En sélectionnant en priorité les clients les plus rapides, l'algorithme permet donc d'accélérer considérablement les cycles d'entraînement fédéré. (Baahmed, Dollinger, Brahmia, & Zghal, 2024)

Le second critère est la diversité des données présentes localement sur chaque appareil. Dans un système d'apprentissage fédéré, chaque client possède des données potentiellement différentes des autres (par exemple, des images de chiffres manuscrits écrits par des personnes différentes, dans le cas du dataset MNIST). En intégrant des clients aux profils de données variés, le modèle global peut généraliser plus efficacement, ce qui se traduit par une meilleure performance globale sur des données jamais vues. (Baahmed, Dollinger, Brahmia, & Zghal, 2024)

Voici comment les deux critères sont calculés dans l'algorithme :

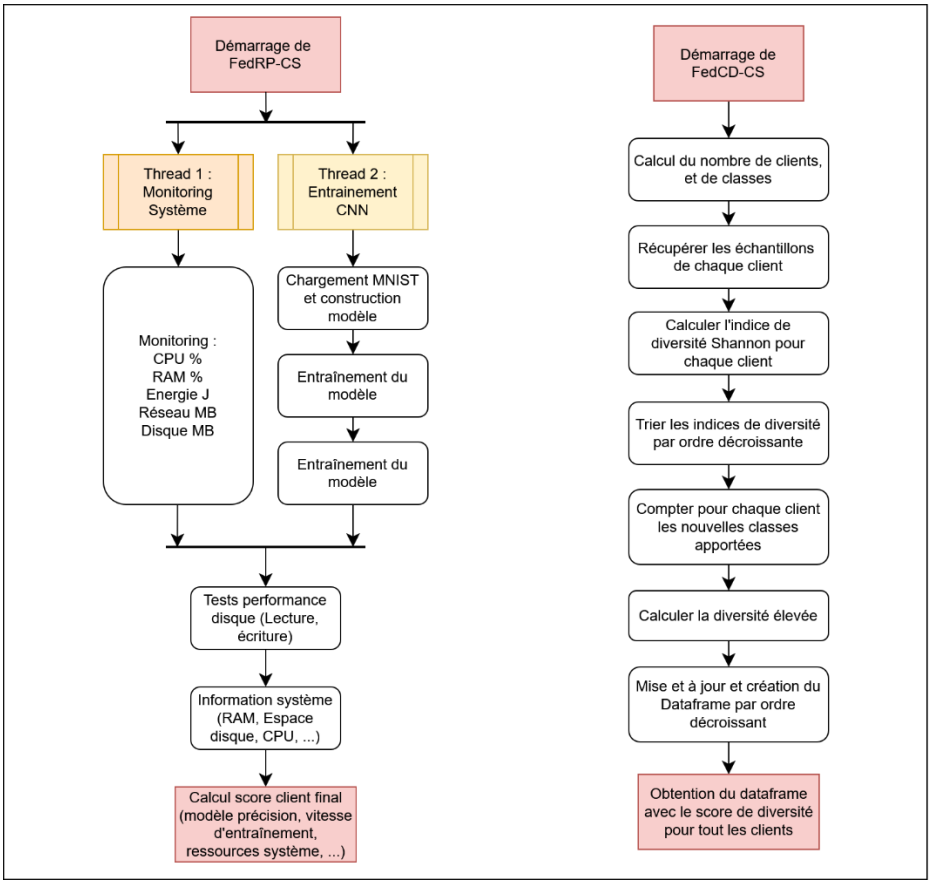


Figure 11 : Schéma des deux algorithmes de sélection de client

Ainsi, j'ai pu utiliser le programme de sélection de client déjà créé par mes tuteurs pour l'adapter à mon système de Federated Learning. En reliant les bonnes fonctions au bon endroit dans le cycle, et en adaptant l'algorithme déjà créé, j'ai pu implémenter cette sélection de client. Maintenant, chaque Raspberry Pi connecté au système calcule et renvoie au serveur un score, qui comprend sa capacité de calcul et la quantité et la diversité statistique de ses données locales. J'ai ensuite développé, côté serveur, une méthode qui utilise ces scores pour sélectionner un sous-ensemble optimal de clients à impliquer dans chaque cycle d'apprentissage. Cette méthode peut choisir les clients, soit de manière entièrement aléatoire, soit en se basant sur leurs scores de performance, leurs scores de diversité, ou une combinaison des deux, selon les besoins du système. Cette approche améliore à la fois l'efficacité du système (en réduisant la latence) et la qualité du modèle fédéré (en maximisant la diversité des contributions).

6.5. Distribution et répartition des données

Pour tester efficacement notre système de Federated Learning, il était essentiel de simuler des conditions réalistes de distribution des données entre les différents clients. Dans l'environnement réel, avec le système final, chaque appareil disposera de capteurs, qui vont généralement créer des données différentes, créant une hétérogénéité naturelle dans la répartition des informations.

Plusieurs approches s'offraient à moi pour reproduire cette variation dans un environnement simulé. Une première option aurait été de répartir les données de manière purement aléatoire entre les clients, ce qui aurait eu l'avantage d'être simple à mettre en place, mais n'aurait pas reflété la complexité et les déséquilibres typiques des cas d'usage réels. Une autre possibilité aurait été d'utiliser des règles mathématiques ou des algorithmes spécifiques, comme le clustering préalable, pour créer des répartitions artificiellement biaisées. Toutefois, ces approches introduisaient des hypothèses arbitraires.

Une solution proposée par mes tuteurs de stage était d'implémenter une distribution de Dirichlet, une solution statistique très robuste permettant de simuler des scénarios avec une bonne hétérogénéité. Cette distribution statistique permet de générer des répartitions de données plus ou moins aléatoires sur chaque client, reflétant ainsi les conditions d'usage réelles où certains appareils peuvent avoir des données très spécialisées tandis que d'autres disposent d'échantillons plus diversifiés.

Cette distribution n'a pas été compliquée à implémenter en Python. Le plus complexe était de bien l'intégrer au reste du programme déjà réalisé, avec les bonnes signalétiques. Il fallait générer cette distribution à chaque début de round, et envoyer les données individuellement à chaque client. Voici, par exemple, une distribution de données qui a été générée lors de mes tests. Les 60 000 images de la base de données sont réparties de manière très fidèle à une distribution aléatoire, sur 50 clients, chacun ayant un nombre de classes (chiffres de 0 à 9) très différent.

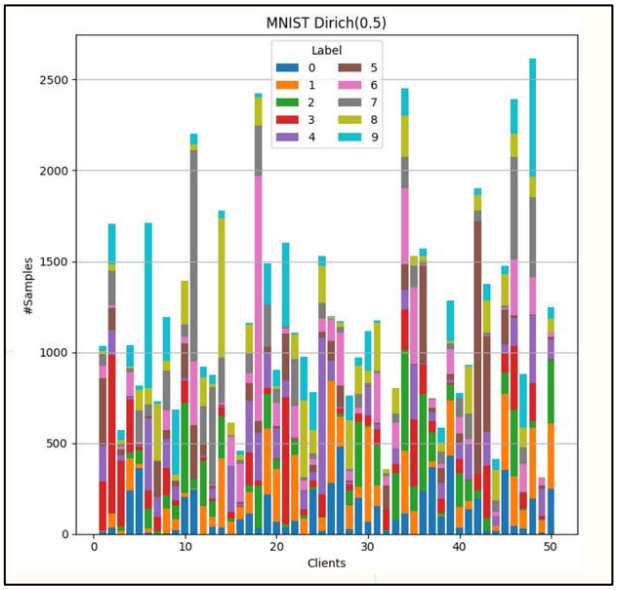


Figure 12 : Distribution Dirichlet générée pendant mes tests

Cette approche de simulation était particulièrement importante dans le contexte de notre projet, car les Raspberry Pi que nous utilisions n'étaient pas encore équipés de capteurs. La distribution de Dirichlet nous a permis de reproduire fidèlement les défis de l'apprentissage fédéré avec des données non uniformément distribuées, préparant ainsi notre système aux conditions réelles d'utilisation.

6.6. Processus d'agrégation

Maintenant que mon programme fonctionnait, avec la communication des clients vers le serveur ainsi que les différents messages de la signalisation, il restait un point crucial, l'agrégation. C'est une étape fondamentale dans le Federated Learning où tous les modèles locaux sont combinés pour former le modèle global amélioré. Sans cette étape, le serveur est incapable d'obtenir une version du modèle commune à tous les clients. Plusieurs techniques d'agrégation existent, chacune présentant des avantages spécifiques selon le contexte, et je vais vous présenter les différentes possibilités qui existent, et vous indiquer l'option que j'ai choisie.

FedAvg (Federated Averaging) est probablement l'approche la plus courante. Cette méthode calcule une moyenne pondérée des paramètres de tous les modèles clients, en tenant compte du nombre d'échantillons d'entraînement de chaque participant. Son principal avantage réside dans sa simplicité d'implémentation et son efficacité prouvée dans de nombreux cas d'usage.

FedProx est une variante qui ajoute des paramètres pour gérer l'hétérogénéité des données et des appareils, et est très utile lorsque certains clients ont des ressources limitées.

FedNova propose une approche normalisée qui s'adapte aux différences de fréquence d'entraînement entre les clients, compensant ainsi les déséquilibres de participation.

Notre choix de FedAvg s'est basé sur plusieurs critères : sa robustesse éprouvée, sa facilité d'implémentation et sa compatibilité avec notre infrastructure. Il est connu, et donc bien documenté, et permet de l'implémenter facilement. En ce qui concerne le modèle mathématique de cette méthode, il correspondait parfaitement à notre cas de simulation. Le processus fonctionne en calculant la moyenne pondérée des poids de chaque couche du réseau de neurones, permettant de créer un modèle global qui bénéficie de l'apprentissage distribué tout en préservant les spécificités apprises par chaque client.

Pour implémenter FedAvg en Python, je me suis basé sur cette formule, qui consiste à agréger les poids des modèles locaux en effectuant une moyenne pondérée selon le nombre d'échantillons utilisés par chaque client durant l'entraînement. Cela garantit que les contributions des clients les plus actifs, en termes de volume de données, sont justement valorisées, tout en conservant une stabilité du modèle global. Cette opération d'agrégation se répète à chaque cycle de communication entre les clients et le serveur, constituant ainsi le cœur battant du processus de Federated Learning dans notre système.

$$w_{t+1} = \sum_{k=1}^{K} \left(\frac{n_k}{n}\right) w_{t+1}^{k}$$

Avec :

t : Numéro du cycle de fédération.

w_{t+1} : Modèle global agrégé à la fin de la t+1-ème cycle de fédération.

K : Nombre total de clients participants dans ce cycle de fédération.

w_{t+1}^{k} : Modèle local entraîné par le client k à la ronde t+1.

n_k : Nombre de données locales du client k.

n : Nombre total de données sur tous les clients.

$\frac{n_k}{n}$: Poids/importance accordée au client k lors de l'agrégation. Ce poids est proportionnel à la taille de son dataset local.

Voici le programme Python que j'ai créé implémentant cette agrégation FedAvg, utilisant la librairie Numpy pour le traitement de données, ainsi que Tensorflow pour la gestion des couches de neurones.

```python
def clients_selection(self, percentage=0.5):
    """Sélectionne un pourcentage de clients pour participer au round.

    Args:
    percentage (float): Pourcentage de clients à sélectionner (0.1 à 1.0).

    Returns:
    list: Liste des clients sélectionnés."""
    clients = self.training_details[self.round_id]["clients"]
    nb_to_select = max(1, round(len(clients) * percentage))
    sorted_clients = sorted(clients.items(), key=lambda item: item[1])
    selected_clients = [ip for ip, _ in sorted_clients[:nb_to_select]]
    return selected_clients
```

6.7. Suivi en temps réel

Pour assurer un suivi efficace du processus d'apprentissage fédéré, j'ai développé un système complet de monitoring en temps réel. J'ai premièrement implémenté un système de logs détaillé qui enregistre dans un fichier chaque étape du processus de Federated Learning : début et fin des phases d'entraînement local, transfert des modèles et résultats d'agrégation. Ces logs permettent de sauvegarder n'importe quel cycle d'apprentissage de l'IA, en analysant les performances en détails.

J'ai également implémenté une interface web qui affiche le nombre de clients actuellement connectés, leur statut individuel (en entraînement, en attente, déconnecté), ainsi que leurs caractéristiques techniques (score de performance, précision du modèle local, …). L'interface permet également de suivre l'évolution de la précision globale du modèle au fil des rounds

d'entraînement. J'ai également ajouté des informations temporelles, comme le temps moyen par round d'entraînement, la durée des phases d'agrégation et la latence de communication.

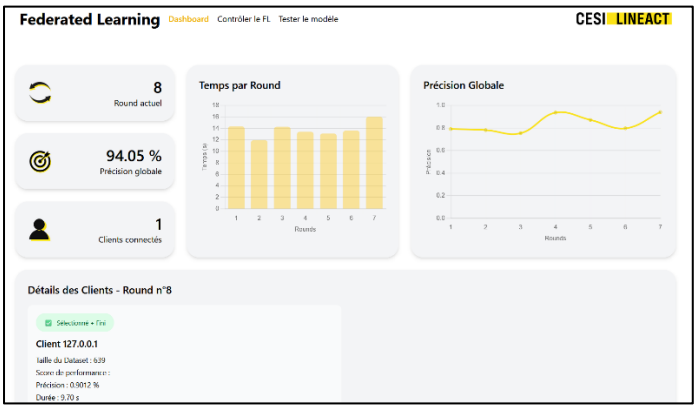


Figure 13-a : Interface web - Dashboard permettant le suivi de toutes les données importantes en temps réel

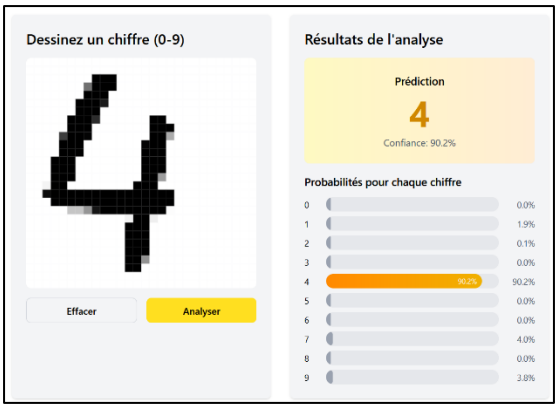


Figure 13-b : Interface web - Analyse de chiffres manuscrits avec le modèle en temps réel

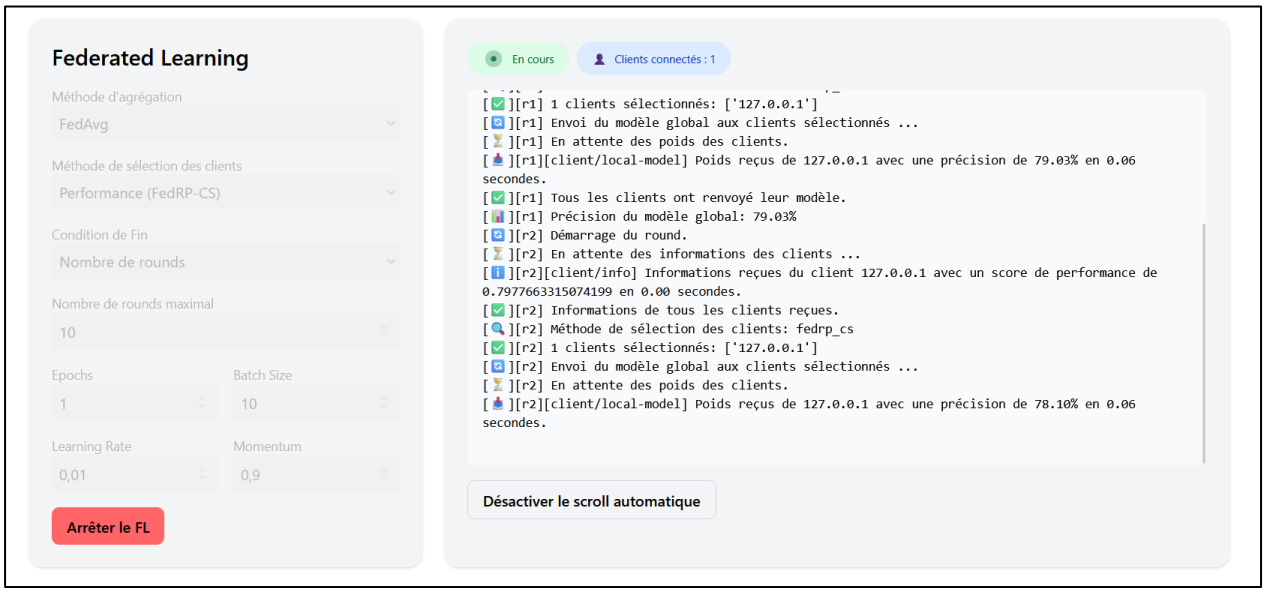


Figure 13-c : Interface Web - Configuration, lancement et logs du Federated Learning en temps réel

Pour implémenter cette interface web, j'ai voulu choisir une technologie adaptée au système déjà mis en place. Pas question de rajouter beaucoup de surcharge au serveur, je voulais quelque chose de très efficace sans impacter le reste du programme. J'ai donc décidé de créer un backend en Python pour rester dans le même écosystème, avec le même langage de programmation. Celui-ci est intégré au système de Federated Learning qui pourra être utilisé par n'importe quel type de frontend avec une API simple. J'ai commencé par faire une comparaison des technologies, en me basant sur ses performances, et notamment le nombre de requêtes JSON qui pouvaient être délivrées. Plusieurs frameworks en Python permettent de créer un backend, comme Django et Flask, mais j'ai finalement choisi FastAPI pour son faible taux d'usage mémoire ainsi que sa rapidité en termes de latence sur les requêtes. Pour le frontend, l'utilisation de Vue.js combinée à TailwindCSS m'a permis de créer des composants dynamiques stylisés facilement. La communication repose sur l'utilisation d'une fonction en TypeScript qui récupère depuis l'IP du serveur backend (FastAPI), avec une requête HTTP, les données en temps réel du système. Celles-ci sont actualisées toutes les secondes.

6.8. Optimisation et tests

J'étais maintenant arrivé à la fin du développement de l'architecture distribuée. L'ensemble du système fonctionnait entre les différents appareils, avec l'agrégation, la simulation des données, la sélection des clients, et même un affichage graphique en temps réel sur une page web pour suivre tout cela. Mais pour en arriver là, et pour gagner encore plus de temps à ce stade, j'ai dû faire beaucoup d'optimisations et je suis passé par plusieurs défis techniques, apparus durant le développement. La synchronisation des clients s'est révélée complexe, notamment lors de la gestion des appareils aux performances hétérogènes. La gestion de la mémoire sur les Raspberry Pi a également été l'un des problèmes majeurs tout au long du développement.

J'ai donc implémenté des techniques de multithreading et de multiprocessing pour optimiser l'utilisation des ressources disponibles. Le serveur central utilise désormais des coroutines avec la librairie asyncio de Python pour gérer simultanément les communications avec plusieurs clients, évitant ainsi les blocages lors des phases de synchronisation. Cette approche permet de traiter les requêtes de manière asynchrone, réduisant considérablement les temps d'attente.

Du côté des clients, j'ai optimisé l'utilisation des ressources de calcul en parallélisant sur plusieurs threads certaines opérations du processus. Cette parallélisation a permis de réduire significativement la durée des phases d'entraînement local, et surtout d'éviter des problèmes de surcharge ou de déconnexion.

Les communications représentant également une partie très importante, j'ai implémenté plusieurs optimisations. J'ai réalisé une compression des modèles avant les transmissions, ce qui a permis de réduire la bande passante, mais aussi la durée de transmission de 30 à 40 % en fonction des appareils. J'ai également implémenté une gestion des reconnexions automatiques pour assurer une certaine robustesse face aux imprévus, qui sont tout de même courants dans un environnement distribué. J'ai utilisé un système de heartbeat qui vérifie régulièrement la connectivité des clients et déclenche automatiquement les procédures de reconnexion si nécessaire.

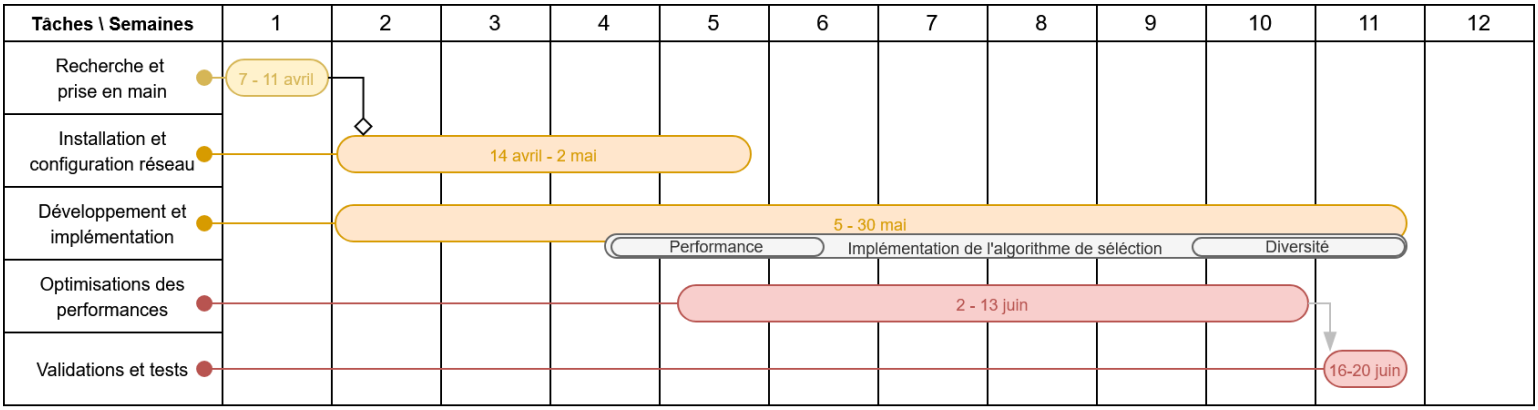
Et pour finir, j'ai réalisé une documentation complète et structurée du code. J'ai pris soin de commenter en détail chaque partie du programme, en expliquant clairement le rôle de chaque fonction, méthode et variable importante, afin de rendre le code aussi compréhensible et réutilisable que possible, tant pour moi que pour d'éventuels futurs contributeurs. En parallèle, j'ai effectué une série de tests sur l'environnement de déploiement en lançant l'ensemble des Raspberry Pi. Cela m'a permis d'observer le comportement du système dans des conditions réelles : les connexions et déconnexions des clients, la synchronisation des échanges, le déroulement de l'agrégation des modèles, ainsi que la stabilité de la communication entre les différentes entités. Ces tests ont également servi à identifier d'éventuels dysfonctionnements ou améliorations à apporter pour fiabiliser encore davantage l'ensemble du système.

VII. Résultats et analyses

Aujourd'hui, plusieurs résultats concrets et fonctionnels ont été obtenus et prouvent l'avancement significatif que j'ai réalisé sur mon projet de Federated Learning. Voici les grandes étapes que j'ai réussi à accomplir jusqu'ici :

- Conception et développement d'un protocole de communication complet, assurant les échanges fiables entre un serveur central et plusieurs clients simulés, déployés sur des Raspberry Pi.
- Mise en place d'un réseau local structuré autour d'un routeur dédié, utilisé par plus de cinq Raspberry Pi, que j'ai installés, configurés et intégrés manuellement au système.
- Développement d'une architecture centrale du système de Federated Learning, en imaginant et codant moi-même toute la logique de signalisation nécessaire à la gestion des cycles d'entraînement distribués.
- Intégration d'un mécanisme sophistiqué de sélection des clients, fondé sur un algorithme conçu par l'équipe de recherche CESI LINEACT, afin d'optimiser le processus collaboratif.
- Ajout d'une méthode de distribution non uniforme des données entre les clients, afin de simuler un environnement réaliste et évaluer le comportement du système en conditions hétérogènes.
- Implémentation d'un algorithme d'agrégation des modèles locaux, ce qui permet d'avoir une cohérence de l'apprentissage global tout en respectant le principe de décentralisation propre au Federated Learning.
- Développement d'une interface web de suivi en temps réel, permettant d'afficher l'état du système, les performances des clients et les étapes d'entraînement de manière claire et interactive.
- Réalisation de diverses optimisations, des batteries de tests et la rédaction d'une documentation complète afin d'assurer la stabilité, la compréhension et la maintenance du projet.

Voici également un planning récapitulatif présentant le détail des missions que j'ai réalisées, ainsi que la chronologie suivie. Il permet de comparer les tâches effectivement accomplies avec le planning prévisionnel que j'avais établi en début de stage à l'aide d'un diagramme de Gantt.



Nous pouvons voir avec ce planning que mon stage s'est déroulé avec un développement, une intégration et un déploiement en continu, avec une méthodologie agile. J'ai apporté des modifications régulières par petites étapes afin d'avoir une adaptation continue tout au long du stage.

Après avoir mené l'ensemble des étapes de préparation, plusieurs séries d'expériences de Federated Learning ont été lancées sur toutes les Raspberry Pi pendant plus de 15 heures. Ces expériences, utilisant différents paramètres, visaient à collecter un maximum d'informations exploitables en vue de les analyser. L'analyse s'est principalement concentrée sur la comparaison des algorithmes de sélection de clients conçus et intégrés dans le système de FL. Deux tableaux permettent d'illustrer les résultats obtenus, chacun en fonction de l'algorithme de sélection utilisé : le premier montre l'évolution de la précision du modèle d'IA au fil du temps, tandis que le second présente la durée d'exécution de chaque round.

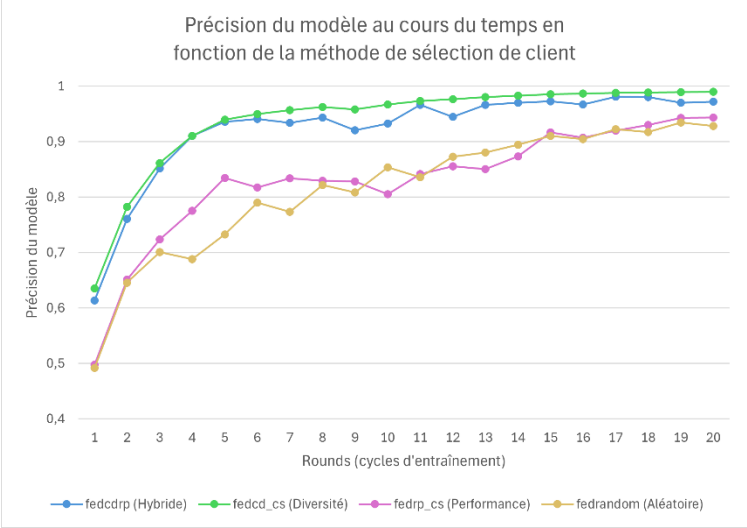


Figure 14-a : Graphique de la précision du modèle au cours du temps

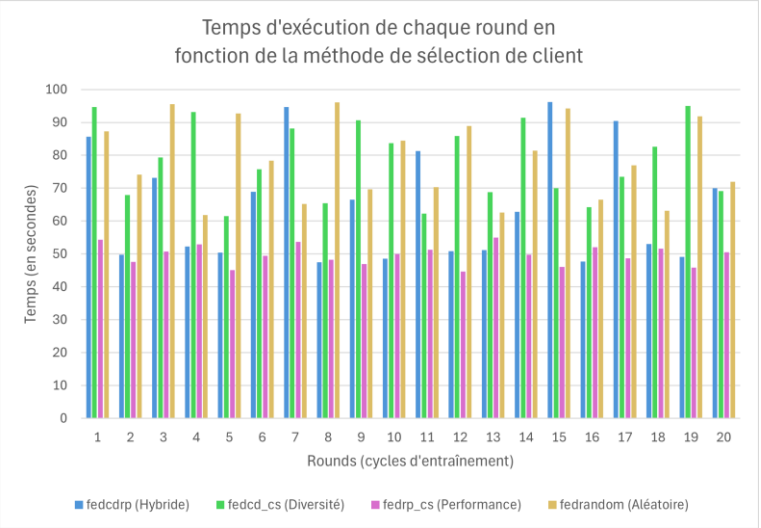


Figure 14-b : Graphique du temps d'exécution de chaque round

Les résultats ont été obtenus en réalisant un entraînement sur 5 Raspberry Pi pendant 20 rounds avec l'algorithme FedAvg et une distribution théorique de Dirichlet sur 50 clients (chaque client disposant en moyenne de 1200 données aléatoires par round). La figure 14-a présente des résultats moyennés sur 10 simulations de FL afin de lisser les courbes.

Sur cette première figure, l'algorithme FedCD, qui tient compte de la diversité des données entre clients, atteint la meilleure précision parmi tous les algorithmes testés, en moyenne plus de 99 %. À l'inverse, l'algorithme FedRP (basé sur la performance des clients) ainsi que la sélection aléatoire offrent des précisions légèrement inférieures. L'algorithme FedCDRP, qui combine diversité et performance, parvient toutefois à maintenir une excellente précision.

La figure 14-b montre que le temps d'exécution varie beaucoup selon l'algorithme. FedCD et la sélection aléatoire, qui ignorent les performances, entraînent avec des durées instables (moyenne de 78 s). À l'inverse, FedRP reste constant autour de 50 secondes, car il sélectionne les clients les plus rapides. FedCDRP alterne entre clients rapides ou plus lents selon les données disponibles, ce qui lui permet d'avoir un bon compromis autour de 64 secondes en moyenne.

J'ai également pu remarquer que l'augmentation du nombre moyen de données par client (en diminuant le nombre de clients théoriques dans la distribution Dirichlet), améliore la précision pour tous les algorithmes, mais allonge aussi les temps d'exécution. Les résultats restent néanmoins cohérents avec les tendances précédemment observées.

Ces résultats permettent de comprendre l'importance du choix des paramètres d'entraînement dans un système de Federated Learning. L'algorithme de sélection de client FedCDRP créé par mes tuteurs, offre un très bon compromis, permettant d'optimiser à la fois la précision du modèle et l'efficacité globale de l'apprentissage.

VIII. Bilan de stage

Ce stage a été pour moi une expérience particulièrement stimulante, enrichissante et humaine. En travaillant au CESI LINEACT, j'ai eu l'opportunité de m'investir dans des sujets passionnants, en lien avec les grandes transformations du monde actuel. J'ai pu développer ma curiosité, approfondir ma réflexion et renforcer mon intérêt pour le domaine de l'intelligence artificielle. Ce stage m'a permis de découvrir une véritable passion que je souhaite désormais cultiver dans mes études.

Cette expérience m'a également offert un cadre de travail idéal pour développer mon autonomie, mon sens de l'organisation et ma rigueur dans la gestion d'un projet aussi complet. J'ai appris à planifier mes tâches avec efficacité, à établir des priorités et à tenir mes objectifs tout en gérant les imprévus. Les échanges réguliers avec mon équipe encadrante, sous forme de réunions hebdomadaires, ont été un soutien précieux, tout en me laissant une grande liberté dans la mise en œuvre de mes solutions. Cette collaboration m'a beaucoup enrichi, aussi bien humainement que professionnellement.

Et pour finir, ce stage m'a offert l'opportunité de découvrir une superbe équipe de recherche, bienveillante, dynamique et passionnée. J'ai eu l'occasion d'échanger avec plusieurs chercheurs et ingénieurs, de découvrir les projets parallèles au mien, portés par d'autres membres de l'équipe, et qui sont tout aussi intéressants et innovants. Cette immersion dans un laboratoire aussi stimulant a été très inspirante, et j'espère désormais avoir la possibilité de poursuivre ce travail dans le cadre d'un futur contrat en alternance au cours des trois prochaines années. Ce serait pour moi un véritable honneur de continuer à évoluer dans cet environnement où curiosité, innovation et esprit d'équipe sont au cœur des préoccupations quotidiennes. Je tiens également à exprimer à nouveau toute ma gratitude envers l'équipe de CESI LINEACT pour son accompagnement et la confiance qu'elle m'a accordée. Ce stage restera une étape marquante de mon parcours, riche en apprentissages et porteuse de belles perspectives pour l'avenir.

VIII. Conclusion

Mon stage au CESI LINEACT avait pour objectifs principaux la mise en place d'une infrastructure distribuée à base de Raspberry Pi et d'un serveur central, l'implémentation d'un système de Federated Learning sur cette architecture, ainsi que le déploiement d'algorithmes de sélection de clients afin d'optimiser les performances globales du système.

À l'issue du stage, l'ensemble de ces objectifs a été atteint. Un système complet a été réalisé, de la phase de conception à l'implémentation, avec des phases d'expérimentation et d'optimisation. Le projet m'a permis d'appliquer concrètement les notions théoriques abordées pendant mes études, tout en relevant des défis techniques exigeants. L'approche évolutive que j'ai appliquée dans le développement de l'architecture laisse également la porte ouverte à de futures améliorations, extensions ou déploiements à plus grande échelle. Pour cette dernière semaine de stage qu'il me reste, mon objectif est d'améliorer les performances globales du programme afin d'accélérer son exécution sur les différents appareils et de renforcer sa résistance aux dysfonctionnements réseau, dans la mesure de mes capacités. Je prévois également dans mes perspectives futures de développer une fonctionnalité de comparaison et de visualisation des simulations sous forme de graphiques, intégrée directement dans l'interface web.

Ce stage a été l'occasion de consolider et d'enrichir l'ensemble des compétences que j'avais acquises au cours de l'année. J'ai notamment renforcé mes connaissances en développement embarqué, en programmation orientée objet, en systèmes et réseaux, ainsi qu'en développement Web. Tous ces savoir-faire ont été mis en œuvre dans le cadre d'un projet à la fois concret, complet et extrêmement formateur.

Pour conclure, ce stage a non seulement validé les objectifs fixés, mais a également permis d'enrichir les projets du laboratoire sur le Federated Learning, tout en posant les bases de futurs travaux.

Bibliographie

Baahmed, A.-R.-E., Dollinger, J.-F., Brahmia, M.-E.-A., & Zghal, M. (2024). *"Hyperparameter Impact on Computational Efficiency in Federated Edge Learning," International Wireless Communications and Mobile Computing (IWCMC).* Récupéré sur ieeexplore.ieee.org: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10592516&isnumber=10592309

BDM. (2024, Novembre Vendredi 8). *ChatGPT, Gemini… combien coûte l'entraînement des modèles d'IA ?* Récupéré sur BDM: https://www.blogdumoderateur.com/chatgpt-gemini-combien-coute-entrainement-modeles-ia/

CESI LINEACT. (s.d.). *Découvrez le laboratoire de recherche.* Récupéré sur CESI LINEACT: https://lineact.cesi.fr/unite-de-recherche/presentation-lineact-cesi/

CPRAM. (2024, Octobre). *Pollution Numérique - IA.* Récupéré sur CPRAM: https://cpram.com/fra/fr/particuliers/publications/megatrends/pollution-numerique-a-la-croisee-des-defis-de-notre-epoque

Flower AI. (s.d.). *Flower Visuals.* Récupéré sur Flower AI: https://flower.ai/visuals/

IBM. (2025, avril 2). *Qu'est-ce que l'apprentissage fédéré ?* Récupéré sur IBM: https://www.ibm.com/fr-fr/think/topics/federated-learning

ISE. (2024, Novembre 26). *Une pollution cachée au coeur de l'innovation.* Récupéré sur Institut Supérieur de l'Environnement: https://institut-superieur-environnement.com/blog/lintelligence-artificielle-une-pollution-cachee-au-coeur-de-linnovation/

Wikipedia. (2017, Décembre 14). *MnistExamples.* Récupéré sur Wikipedia: https://commons.wikimedia.org/wiki/File:MnistExamples.png

Wikipedia. (s.d.). *Apprentissage fédéré.* Récupéré sur Wikipedia: https://fr.wikipedia.org/wiki/Apprentissage_f%C3%A9d%C3%A9r%C3%A9

Wikipedia. (s.d.). *Base de données MNIST.* Récupéré sur Wikipedia: https://fr.wikipedia.org/wiki/Base_de_donn%C3%A9es_MNIST

Glossaire

[1] Architecture distribuée : Système composé de plusieurs appareils interconnectés, qui coopèrent pour exécuter des tâches. Chaque nœud peut traiter localement une partie des données ou des calculs.

[2] Apprentissage fédéré (Federated Learning ou FL) : Méthode d'apprentissage automatique décentralisée où plusieurs appareils (ou clients) entraînent localement un modèle, puis envoient les poids mis à jour à un serveur central qui les agrège sans accéder aux données locales.

[3] Raspberry Pi : Mini-ordinateur monocarte, disponible facilement et à bas coût, souvent utilisé dans des projets IoT ou embarqués.

[4] Modèle : Structure mathématique entraînée à partir de données pour effectuer des prédictions, classifications ou prises de décision. Il capture les relations entre les entrées et les sorties observées durant l'entraînement.

[5] Intelligence Artificielle (IA) : Systèmes capables d'imiter certaines capacités humaines comme apprendre, comprendre ou décider. Utilisée pour reconnaître des images, traduire du texte, ou jouer à des jeux, par exemple.

[6] Client : Appareil (ordinateur, smartphone, Raspberry Pi, ...) qui détient des données locales et participe à l'entraînement du modèle en effectuant des calculs sur ces données.

[7] Apprentissage automatique (Machine Learning) : Branche de l'IA qui consiste à développer des algorithmes permettant à un système d'apprendre automatiquement à partir de données, sans être explicitement programmé pour chaque tâche.

[8] Apprentissage profond (Deep Learning) : Sous-catégorie de l'apprentissage automatique utilisant des réseaux de neurones profonds (composés de plusieurs couches) pour traiter des données complexes comme des images, du texte ou de l'audio.

[9] Neurone artificiel : Unité de calcul inspirée du neurone biologique. Il reçoit des entrées pondérées, applique une fonction d'activation, et produit une sortie. C'est le composant de base des réseaux de neurones.

[10] Réseau de neurones (Neural Network) : Ensemble de neurones artificiels organisés en couches (entrée, cachées, sortie). Il apprend à réaliser des tâches en ajustant les poids entre les neurones selon les données d'entraînement.

[11] Poids : Paramètres numériques ajustables dans un réseau de neurones. Ils déterminent l'importance de chaque entrée dans les calculs effectués par un neurone. Leur ajustement progressif constitue l'apprentissage du modèle.

[12] Apprentissage centralisé : Méthode d'apprentissage automatique où toutes les données sont regroupées dans un serveur ou un centre de calcul unique. Le modèle est entraîné globalement à partir de cet ensemble de données centralisées.

[13] Agrégation : Processus par lequel le serveur central combine les modèles (ou poids) mis à jour par les clients pour obtenir un modèle global amélioré, sans jamais accéder aux données locales.

[14] Straggler Problem : Situation où certains clients dans un système distribué mettent beaucoup plus de temps que les autres à terminer leurs tâches (ex. entraînement local), ralentissant ainsi le processus global d'agrégation ou d'apprentissage.