



Rapport intermédiaire du Projet 5A

Détection des voix frauduleuses à l'aide du modèle AASIST et intégration MLOps

Réalisé par :

Youssef ERMADI

Ikram HADDOUCH

Hamza EL ACHHAB

Membres de jury :

- Mr. Aghilas Sini

- Mr. Shamsi Meysam

- Mr. MAY Madeth

Remerciements

Nous souhaitons exprimer notre sincère gratitude à **M. Sini Aghilas** et **M. Meysam Shamsi**, qui nous ont accompagnés tout au long de ce projet. Leur encadrement bienveillant, leur expertise technique et leurs conseils éclairés ont été d'une grande valeur pour notre travail en groupe.

Leur disponibilité, leur pédagogie et leur capacité à répondre à nos interrogations avec patience nous ont permis de progresser face aux défis rencontrés. Grâce à leurs orientations, nous avons pu approfondir nos compétences techniques, affiner nos méthodologies et mener ce projet avec rigueur et efficacité.

Leur soutien a favorisé une dynamique de groupe productive et collaborative, renforçant ainsi notre esprit d'équipe et notre capacité à résoudre collectivement les problématiques complexes. Nous leur sommes reconnaissants pour la qualité de leur encadrement et pour leur rôle clé dans la réussite de cette expérience enrichissante.

Enfin, leur implication constante a été une source d'inspiration qui nous a encouragés à donner le meilleur de nous-mêmes. Nous leur adressons nos plus sincères remerciements.

Abstract

This project delves into the application of MLOps methodologies to design and implement a robust framework for detecting audio spoofing. By leveraging the **AASIST model**, an advanced architecture built upon deep learning principles, we developed an API capable of efficiently analyzing and classifying audio files to determine their authenticity.

The project encompasses multiple stages: preprocessing audio data to ensure quality and relevance, training the AASIST model with carefully curated datasets, integrating the trained model into an API, and deploying the solution in a scalable and production-ready environment. Emphasis was placed on automating workflows through the implementation of end-to-end pipelines, ensuring repeatability, traceability, and efficiency across the system.

MLflow was utilized as a central tool for managing experiments, tracking model versions, and facilitating collaborative development. The solution also integrates monitoring features to ensure reliability post-deployment. By addressing real-world challenges, this project showcases how MLOps practices can transform machine learning prototypes into reliable and scalable production-grade systems, with a particular focus on audio-based applications.

Résumé

Ce projet explore l'application des méthodologies MLOps pour concevoir et mettre en œuvre un cadre robuste permettant de détecter les falsifications dans les fichiers audio. En s'appuyant sur le modèle **AASIST**, une architecture avancée basée sur les principes de l'apprentissage profond, nous avons développé une API capable d'analyser et de classer efficacement les fichiers audio selon leur authenticité.

Le projet se divise en plusieurs étapes : prétraitement des données audio pour garantir leur qualité et leur pertinence, entraînement du modèle AASIST à l'aide de jeux de données soigneusement sélectionnés, intégration du modèle dans une API, et déploiement de la solution dans un environnement évolutif et prêt pour la production. Une attention particulière a été accordée à l'automatisation des workflows grâce à la mise en place de pipelines de bout en bout, garantissant la répétabilité, la traçabilité et l'efficacité du système.

L'utilisation d'**MLflow** a permis de centraliser la gestion des expériences, le suivi des versions des modèles et la collaboration entre les membres de l'équipe. La solution intègre également des fonctionnalités de surveillance pour assurer sa fiabilité après déploiement. En répondant à des problématiques réelles, ce projet illustre comment les pratiques MLOps peuvent transformer des prototypes d'apprentissage automatique en systèmes fiables et évolutifs, avec un accent particulier sur les applications basées sur l'audio.

Introduction général

Avec l'émergence des technologies de deepfake, les méthodes de falsification vocale deviennent de plus en plus sophistiquées, rendant les systèmes traditionnels de détection obsolètes. Il devient alors impératif de développer des solutions robustes, capables d'identifier ces tentatives de fraude avec un haut degré de précision. C'est dans ce contexte que le projet présenté dans ce rapport prend tout son sens.

L'objectif de ce projet est de concevoir et de déployer un système performant et évolutif pour la détection des voix frauduleuses, en s'appuyant sur le modèle AASIST, une architecture avancée de machine learning spécialisée dans la détection de falsifications audio. Ce modèle est intégré dans une API web, permettant une accessibilité et une scalabilité optimales. Afin d'assurer une gestion efficace de l'ensemble du cycle de vie des modèles, les méthodologies MLOps ont été adoptées, offrant des garanties en termes de reproductibilité, d'automatisation et de suivi des performances.

Ce rapport détaille les différentes étapes de réalisation de ce projet, depuis la compréhension des enjeux et la structuration des travaux, jusqu'au développement technique et à l'intégration du modèle dans un environnement de production. Il met également en lumière les défis rencontrés, les solutions apportées et les perspectives pour étendre et améliorer ce système à l'avenir. À travers cette démarche, nous souhaitons démontrer comment les pratiques modernes d'ingénierie et de gestion des projets ML peuvent être mobilisées pour répondre à des besoins concrets et urgents dans un monde de plus en plus numérique et interconnecté.

Table de matière

Remerciements.....	1
Abstract.....	2
Résumé.....	2
Introduction général.....	3
Table de matière.....	4
Chapitre 1 : Introduction.....	5
1. Contexte général : De DevOps à MLOps.....	5
2. Problématique : Les contraintes du passage de DevOps à MLOps.....	6
3. Objectifs : Pourquoi passer à MLOps ?.....	9
4 Contexte du projet : Cas applicatif - ASVspoof.....	10
4.1 Présentation d'ASVspoof.....	10
4.2 Objectif du projet dans le cadre MLOps.....	10
4.3 Limites et défis spécifiques.....	10
Chapitre 2 : Organisation et outils.....	12
1. Objectif du chapitre.....	12
2. Organisation du projet.....	12
3. Communication entre équipes.....	13
4. Cahier des charges.....	13
5. Planning et gestion du temps.....	13
6. Outils et technologies.....	14
Chapitre 3 : Application.....	15
1. Introduction.....	15
2. Machine Learning Engineering.....	15
a. Modèle AASIST.....	15
b. Architecture du Modèle.....	16
c. Tests du modèle.....	17
❖ Préparation des fichiers de test.....	17
❖ Fonctionnement du test du modèle.....	17
❖ Résultats :.....	18
3. Coding Engineering.....	19
a. Création de l'API.....	19
b. Choix techniques pour l'API.....	19
c. Structure de l'API.....	20
d. Développement et Implémentation de l'API.....	20
e. Tester API dans Insomnia.....	20
4. Conclusion.....	21
Chapitre 4 : Conclusion et perspectives.....	22
1. Résumé du projet.....	22
2. Bilan du travail effectué.....	22
3. Perspectives.....	22
Conclusion Générale.....	24
Annexes.....	25
Annexe 1.....	25
Annexe 2.....	25
Références :.....	27

Chapitre 1 : Introduction

1. Contexte général : De DevOps à MLOps

DevOps : Un paradigme de collaboration et d'efficacité

Le concept de **DevOps** est né de la nécessité de renforcer la collaboration entre les équipes de développement logiciel (Dev) et celles en charge des opérations informatiques (Ops). Il vise à briser les silos organisationnels et à promouvoir des pratiques agiles pour accélérer le développement, le déploiement et la maintenance des logiciels. Parmi les principes fondamentaux de DevOps, on retrouve :

- **Intégration continue (CI)** et **livraison continue (CD)** pour automatiser le développement et le déploiement.
- **Monitoring** en continu pour optimiser les performances et détecter rapidement les anomalies.
- Une culture d'amélioration continue favorisant l'innovation tout en minimisant les risques.

Grâce à DevOps, les entreprises ont pu atteindre des cycles de développement plus courts et améliorer leurs produits et services tout en maintenant une qualité élevée.

La transition vers MLOps

Avec l'adoption croissante de l'intelligence artificielle et du machine learning dans divers secteurs, une nouvelle dimension s'est ajoutée aux défis déjà adressés par DevOps : la gestion des modèles ML et des données associées. Cette transition donne naissance au paradigme **MLOps** (*Machine Learning Operations*), une extension des pratiques DevOps, spécifiquement adaptée aux besoins du cycle de vie des modèles ML.

Puisque le MLOps est ainsi une extension des principes du DevOps aux enjeux du machine learning, les principes généraux sont les mêmes que ceux évoqués précédemment mais ceux-ci s'adaptent à la problématique de la gestion du cycle de vie d'un modèle:

1. **la reproductibilité** : les résultats de chaque expérimentation, fructueuse comme infructueuse, doivent pouvoir être reproduits sans coût. Cela implique d'abord une certaine rigueur dans la gestion des packages, la gestion des environnements, la gestion des librairies système, le contrôle de version du code, etc.
2. **le contrôle de version**: au-delà du simple suivi des versions du code, pour reproduire de manière identique les résultats d'un code c'est l'ensemble des inputs et paramètres influençant l'entraînement d'un modèle (données d'entraînement, hyper-paramètres, etc.) qui doivent être versionnées avec le modèle.
3. **l'automatisation** : afin de favoriser les boucles d'amélioration continue, le cycle de vie du modèle (tests, build, validation, déploiement) doit être automatisé au maximum. Les outils issus de l'approche DevOps, en particulier l'intégration et déploiement continus (CI/CD), doivent être mis en place.

4. **Collaboration** : valoriser une culture de travail collaborative autour des projets de ML, dans laquelle la communication au sein des équipes doit permettre de réduire le travail en silos et bénéficier des expertises des différents métiers parti prenantes d'un modèle (analystes, data engineers, devs...). Sur le plan technique, les outils MLOps utilisés doivent favoriser le travail collaboratif sur les données, le modèle et le code utilisés par le projet.
5. **L'amélioration continue** : une fois déployé, il est essentiel de s'assurer que le modèle fonctionne bien comme attendu en évaluant ses performances sur des données réelles à l'aide d'outils de monitoring en continu. Dans le cas d'une dégradation des performances dans le temps, un ré-entraînement périodique ou un entraînement en continu du modèle doivent être envisagés.

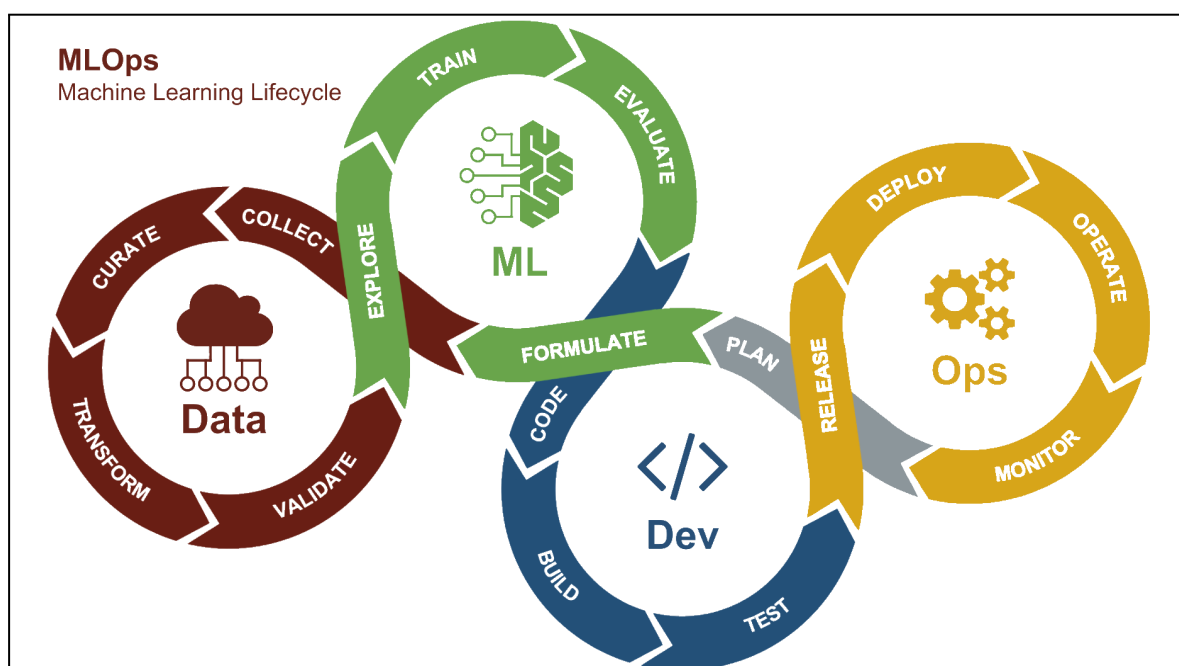


Figure 1: MLOps Lifecycle

2. Problématique : Les contraintes du passage de DevOps à MLOps

Limitations de DevOps pour les projets de machine learning

Bien que DevOps a révolutionné le développement et la maintenance des logiciels traditionnels, ses pratiques et outils montrent rapidement leurs limites lorsqu'il s'agit de projets basés sur le machine learning. Ces limitations sont principalement dues à la nature même des modèles ML et à leur dépendance aux données :

- a. **Le code seul ne suffit pas** : Contrairement au développement logiciel classique, où le code est la principale unité de production, les modèles ML s'appuient fortement sur des données d'entraînement. Ces données évoluent dans le temps (phénomène de *data drift*), ce qui peut altérer la performance des modèles en production.

- b. **Automatisation complexe** : Dans DevOps, les pipelines CI/CD sont bien établis pour tester et déployer du code. Cependant, en MLOps, ces pipelines doivent inclure des étapes supplémentaires, comme :
- La validation des données.
 - Le réentraînement des modèles.
 - La gestion des versions des données et des modèles.
- c. **Dépendances croissantes** : Les modèles ML dépendent non seulement du code, mais aussi des hyperparamètres, des configurations, et des environnements d'exécution, ce qui complexifie leur gestion dans des environnements DevOps classiques.

Différences clés entre DevOps et MLOps

Le passage de DevOps à MLOps nécessite de repenser certaines pratiques pour s'adapter aux spécificités des projets ML. Voici les principales différences :

Aspect	DevOps	MLOps
Unité principale	Code	Modèles + Données
Tests automatisés	Test du code (unitaires, intégration)	Validation des données et des modèles
contrôle de version	Code source	Modèles, données, pipelines
Pipeline CI/CD	Déploiement de logiciels	Réentraînement, validation, déploiement des modèles
Maintenance	Suivi des logs et des performances	Monitoring des modèles et data drift

Adoption d'outils et de nouvelles pratiques en MLOps

Pour répondre aux défis spécifiques de MLOps, des outils dédiés et des pratiques innovantes sont nécessaires. Ces outils permettent d'automatiser et de standardiser les processus, en offrant des solutions adaptées à chaque étape du cycle de vie d'un modèle ML :

- a. **Outils spécifiques** :

MLFlow : Gestion des expériences, des versions de modèles et des déploiements.

Kubeflow : Orchestration de pipelines ML, intégration avec Kubernetes pour la scalabilité.

b. **Nouvelles pratiques :**

Monitoring des modèles : Surveiller les performances des modèles en production pour détecter le *data drift* ou les biais.

Gestion des pipelines de données : Automatiser la collecte, le nettoyage et la transformation des données pour garantir leur qualité.

Automatisation du réentraînement : Mettre en place des pipelines qui détectent les besoins de réentraînement basés sur des seuils prédéfinis.

Servir un modèle ML à des utilisateurs

Une partie très importante des projets de machine learning est la mise à disposition des modèles entraînés à d'autres utilisateurs. Une manière triviale de transmettre le modèle serait de partager le code et toutes les informations nécessaires pour qu'une autre personne ré-entraîne le modèle de son côté. Évidemment, ce procédé n'est pas optimal, car il suppose que tous les utilisateurs disposent des ressources, infrastructures et des connaissances nécessaires pour réaliser l'entraînement.

L'objectif est donc de mettre à disposition le modèle de manière simple et efficace. Alors dans le cadre de ce projet, nous avons choisi d'utiliser une API REST pour mettre à disposition un modèle de machine learning. Cela nous semble être la méthode la plus adaptée dans une grande majorité des cas, car elle répond à plusieurs critères :

- **Simplicité** : les API REST permettent de créer une porte d'entrée qui peut cacher la complexité sous-jacente du modèle, facilitant ainsi sa mise à disposition.
- **Standardisation** : l'un des principaux avantages des API REST est qu'elles reposent sur le standard HTTP. Cela signifie qu'elles sont agnostiques au langage de programmation utilisé et que les requêtes peuvent être réalisées en XML, JSON, HTML, etc.
- **Modularité** : le client et le serveur sont indépendants. En d'autres termes, le stockage des données, l'interface utilisateur ou encore la gestion du modèle sont complètement séparés de la mise à disposition (le serveur).

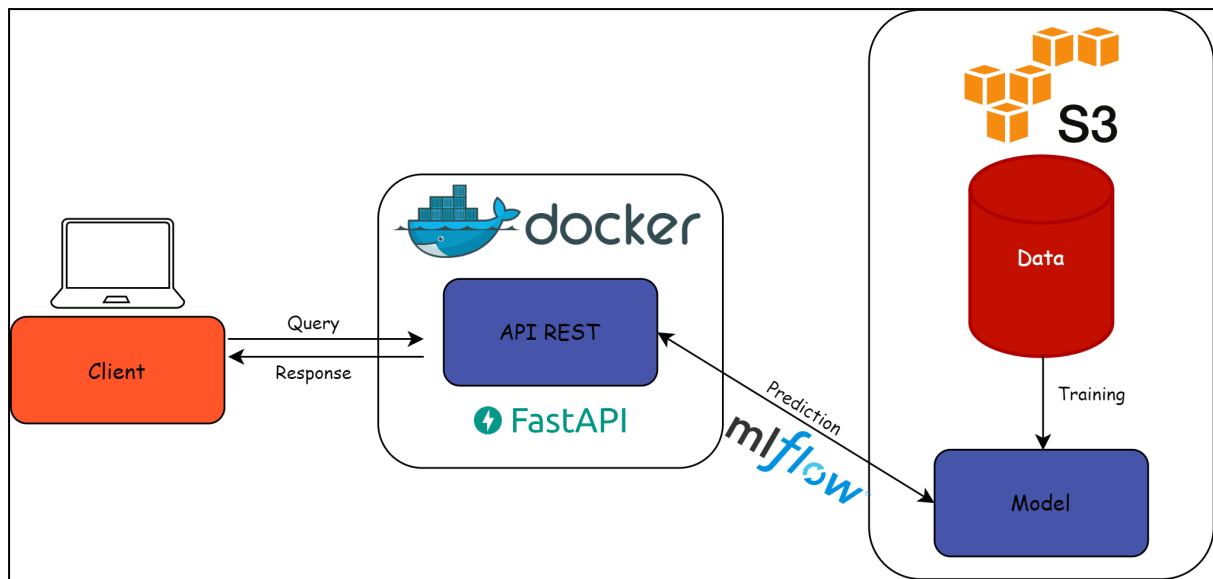


Figure 2: Exposer un modèle de ML via une API

3. Objectifs : Pourquoi passer à MLOps ?

Résoudre les problèmes de scalabilité et de répliquabilité

L'un des principaux objectifs de MLOps est de résoudre les défis spécifiques liés à la **scalabilité** et à la **répliquabilité** des projets de machine learning. En effet :

- **Scalabilité** : Les projets ML doivent pouvoir évoluer pour traiter des volumes croissants de données et répondre à des besoins plus complexes. MLOps permet d'automatiser des pipelines de traitement et de déploiement, garantissant une montée en charge fluide.
- **Répliquabilité** : Pour garantir la cohérence des résultats, il est essentiel de pouvoir reproduire les expériences et les prédictions dans différents environnements. Grâce à la gestion des versions des modèles, des données et des pipelines, MLOps facilite cette répliquabilité.

Garantir une collaboration fluide entre les équipes

MLOps favorise une **collaboration fluide** entre les différentes parties prenantes du projet, notamment :

- **Les data scientists** : Responsables de l'entraînement et de l'optimisation des modèles.
- **Les ingénieurs DevOps et ML** : Chargés de déployer, surveiller et maintenir les modèles en production.
- **Les équipes métiers** : Qui s'appuient sur les résultats pour prendre des décisions.

4 Contexte du projet : Cas applicatif - ASVspoof

4.1 Présentation d'ASVspoof

ASVspoof (**Automatic Speaker Verification Spoofing**) est un projet scientifique et technique visant à répondre à une problématique grandissante : la détection et la prévention des attaques de spoofing dans les systèmes de vérification vocale automatique. Ces attaques, qui consistent à falsifier des enregistrements vocaux pour usurper l'identité d'une personne, posent des défis importants dans des secteurs tels que :

- **La finance** : Fraude dans les systèmes bancaires basés sur la reconnaissance vocale.
- **Les services clients** : Usurpation d'identité pour accéder à des informations sensibles.
- **Les plateformes de sécurité** : Compromission des systèmes d'authentification vocale.

Avec l'avènement des technologies d'intelligence artificielle, telles que les deepfakes vocaux, la sophistication des attaques ne cesse d'augmenter, rendant les systèmes traditionnels de détection obsolètes.

4.2 Objectif du projet dans le cadre MLOps

Dans ce contexte, le projet propose de développer une **plateforme web** intégrant un modèle de machine learning avancé, basé sur l'architecture **AASIST**, pour détecter efficacement les voix frauduleuses. Les principaux objectifs sont :

1. **Détection précise** : Mettre en place un système capable de distinguer les enregistrements authentiques des voix falsifiées, avec un haut degré de fiabilité.
2. **Automatisation via MLOps** : Intégrer les pratiques MLOps pour assurer une gestion efficace du cycle de vie du modèle, comprenant :
 - Le traitement des données audio.
 - L'entraînement et le réentraînement des modèles.
 - Le déploiement et la surveillance des performances en production.
3. **Accessibilité et scalabilité** : Fournir un système accessible via une API web, qui soit capable de s'adapter aux besoins des utilisateurs, même à grande échelle.

4.3 Limites et défis spécifiques

Le projet ASVspoof, bien qu'ambitieux, rencontre plusieurs **défis spécifiques** liés à sa nature innovante :

1. **Qualité des données audio** : Les performances du modèle dépendent fortement de la qualité et de la représentativité des données d'entraînement. Des bruits ou des variations dans les enregistrements peuvent affecter la précision des prédictions.

2. **Évolution des attaques** : Les techniques de spoofing évoluent rapidement, nécessitant une mise à jour fréquente des modèles pour rester efficace face aux nouvelles menaces.
3. **Complexité des modèles** : L'intégration d'architectures avancées, comme AASIST, demande des ressources importantes pour l'entraînement et la maintenance, ainsi qu'une expertise technique pour leur optimisation.
4. **Déploiement en production** : Garantir des prédictions rapides et fiables, tout en gérant les contraintes de scalabilité et de monitoring des modèles en production.

Chapitre 2 : Organisation et outils

1. Objectif du chapitre

Ce chapitre présente le rôle de l'organisation et du cahier des charges dans la réussite du projet. Il met en évidence la structuration des équipes, les outils et méthodes de gestion employés, ainsi que la planification et la communication mises en place. L'objectif est de montrer comment une bonne organisation, soutenue par des outils adaptés, facilite la coordination des tâches et l'atteinte des objectifs du projet.

2. Organisation du projet

Le projet s'est déroulé en plusieurs phases, chacune organisée autour de tâches spécifiques réparties entre les membres de l'équipe, permettant une progression structurée et collaborative :

Première phase : Développement de l'interface (Frontend)

- Deux membres de l'équipe se sont concentrés sur le développement de l'interface utilisateur. Cette interface permet d'enregistrer ou de téléverser des fichiers audio.
- Le troisième membre s'est chargé de la documentation sur les concepts liés à l'ASV (Automatic Speaker Verification) et au CM (Countermeasure) pour mieux comprendre le contexte technique et réaliser des schémas explicatifs pour la suite du projet.

Deuxième phase : Mise en place d'une API simple

- Une API basique a été développée pour recevoir un fichier audio en entrée et appliquer une modification simple, avant de renvoyer le fichier modifié.
- Un membre s'est occupé de la création de cette API, un autre a réalisé les tests via l'outil **Insomnia**, et le troisième a intégré l'API avec le frontend.
- Cette intégration a permis de relier l'interface utilisateur à l'API, offrant ainsi la possibilité d'enregistrer ou de téléverser un fichier audio via le frontend et de recevoir l'audio modifié en retour.

Troisième phase : Test et utilisation du modèle AI AASIST

- Le modèle **AASIST**, préalablement entraîné et fourni par les encadrants, a été testé pour évaluer sa capacité à détecter les voix authentiques et frauduleuses.
- Un membre de l'équipe a mené les tests sur des fichiers audio contenant des voix frauduleuses et authentiques.

- Les deux autres membres ont développé une nouvelle API qui intègre le modèle **AASIST** pour analyser les fichiers audio et déterminer leur nature (authentique ou frauduleuse). Cette API a également été testée avec **Insomnia** pour garantir son bon fonctionnement.

3. Communication entre équipes

Une communication fluide et régulière a été maintenue afin d'assurer une cohésion au sein de l'équipe. Pour cela, plusieurs méthodes ont été employées :

- **Réunions hebdomadaires** : Un point régulier a permis de faire le bilan de l'avancement, de discuter des difficultés rencontrées et de planifier les étapes suivantes.
- **Outils de communication** : L'utilisation de Microsoft Teams a facilité le partage instantané d'informations, la discussion autour des problèmes techniques et la prise de décisions rapides.
- **Partage des ressources** : Les codes sources, documents ont été centralisés sur une plateforme de versionnement (GitLab) et sur le site de l'université dédié à la gestion de projets. Ainsi, chaque membre pouvait facilement accéder aux dernières mises à jour, suivre l'avancement et effectuer des retours.

4. Cahier des charges

Le cahier des charges a servi de référence, définissant clairement les objectifs, les fonctionnalités et les contraintes du projet. Il a permis de cadrer les travaux, d'éviter les dérives et de guider les choix techniques.

- **Objectifs du projet** : L'objectif principal consistait à détecter les fichiers audio « spoof » (frauduleux) à l'aide du modèle AASIST, via une API fonctionnelle intégrée dans une plateforme web. Cela inclut la capacité d'enregistrer ou de téléverser des fichiers audio, de les transmettre au système de détection, puis de retourner un résultat indiquant si l'enregistrement était authentique ou frauduleux.

5. Planning et gestion du temps

a. **Diagramme de Gantt** : Un diagramme de Gantt a été établi pour détailler les principales étapes, du développement du frontend à l'intégration du modèle AASIST. Il a permis de visualiser les dépendances entre tâches, d'estimer la durée de chaque étape et de répartir efficacement la charge de travail.

b. **Suivi des tâches** : Le suivi des tâches a été assuré grâce au site de gestion de projet de l'université, permettant ainsi aux encadrants et à l'équipe de suivre l'évolution du projet.

6. Outils et technologies

Plusieurs outils ont été employés pour soutenir le développement et la gestion du projet :

- **Versioning** : GitLab a été utilisé pour centraliser le code du modèle AI.
- **Développement** : Le backend a été implémenté avec Python et FastAPI, tandis que le frontend a fait appel à HTML, CSS et JavaScript. Pour le traitement et les tests du modèle, Google Colab a été utilisé. Visual Studio Code a servi d'éditeur principal.
- **Tests** : L'outil Insomnia a facilité la vérification et la validation de l'API, assurant que chaque fonctionnalité répondait aux spécifications.

Chapitre 3 : Application

1. Introduction

Dans le cadre de notre projet, nous avons structuré notre travail en deux axes principaux propres au paradigme MLOps :

1. **Machine Learning Engineering** : Préparation et intégration du modèle AASIST pour la détection de spoofing audio.
2. **Coding Engineering** : Création d'une API permettant une utilisation fonctionnelle, automatisée et scalable de ce modèle.

Contexte : Explique l'importance de la détection des attaques de spoofing dans les systèmes de vérification vocale automatique (ASV). Mentionne les défis posés par les algorithmes avancés de spoofing et l'importance de développer des solutions robustes et efficaces.

L'objectif principal de ce chapitre est de présenter les travaux réalisés pour concevoir une API capable d'analyser des fichiers audio et de fournir des résultats précis, accompagnés de logs récapitulatifs.

2. Machine Learning Engineering

a. Modèle AASIST

Le modèle AASIST (**Audio Anti-Spoofing using Integrated Spectrogram and Temporal information**) est basé sur l'apprentissage profond. Il est conçu pour détecter les attaques de spoofing dans les fichiers audio, en se basant sur des caractéristiques spectrales et temporelles.

- **Entrée** : Fichiers audio au format WAV ou MP3.
- **Traitement** : Analyse spectro-temporelle via un réseau de neurones convolutifs (CNN) pour extraire les caractéristiques des données audio.
- **Sortie** : Classification en deux catégories :
 - **Spoof** : Audio falsifié.
 - **BonaFide**: Audio authentique.Un pourcentage de confiance accompagne chaque prédiction, offrant une indication sur la fiabilité du résultat.

b. Architecture du Modèle

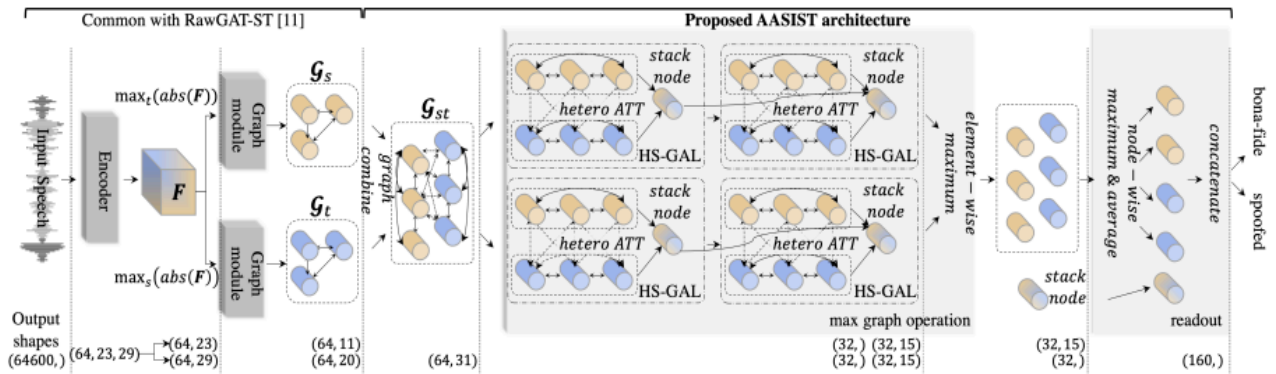


Figure 3 : Architecture du Modèle AASIST

Le fonctionnement de l'AASIST est structuré comme suit (Fig. 3) :

- **Extraction des caractéristiques :**
 - Le modèle reçoit en entrée des fichiers audio bruts (*waveforms*) au format **WAV** ou **MP3**.
 - Un encodeur basé sur **RawNet2** extrait un vecteur de caractéristiques intermédiaires noté **F**.
 - L'encodeur inclut des couches de convolution sinc et des blocs résiduels.
- **Analyse spectro-temporelle :**
 - Deux modules de graphes analysent **F** en parallèle :
 - **Module temporel** : Capture les relations temporelles dans l'audio.
 - **Module spectral** : Analyse des relations entre fréquences.
 - Ces deux modules sont essentiels pour capturer des aspects complémentaires des données.
- **Fusion des graphes hétérogènes :**
 - Les sorties des modules sont combinées en un **graphe hétérogène**.
 - Deux branches parallèles appliquent des **couches d'attention spectro-temporelle hétérogène (HS-GAL)** et des **graph pooling layers**, réduisant progressivement la taille des graphes.
- **Max Graph Operation :**
 - Les deux branches fusionnent à l'aide d'une opération de **maximum élémentaire**, qui conserve les caractéristiques les plus pertinentes.
- **Schéma de lecture (Readout) :**
 - Une représentation dense est construite en combinant le maximum, la moyenne des nœuds, et un **nœud empilé**.
- **Classification finale :**
 - La couche de sortie classe l'audio comme **Bonafide** ou **Spoof**, avec un pourcentage de confiance.

c. Tests du modèle

Pour évaluer la performance du modèle AASIST, nous avons utilisé le **dataset ASVspoof 2019**, qui est un référentiel bien connu pour l'évaluation de systèmes de détection de spoofing audio. Le modèle a été testé à l'aide de plusieurs échantillons audio dans le but de valider sa capacité à détecter des attaques de type spoofing, tout en vérifiant la robustesse de l'algorithme avant son intégration dans l'API.

Les tests ont été réalisés en appliquant un processus d'inférence sur des fichiers audio à l'aide d'un modèle pré-entraîné, tout en consignnant les résultats dans un fichier de log pour analyse.

❖ Préparation des fichiers de test

Avant de procéder aux tests du modèle, plusieurs étapes de prétraitement ont été appliquées aux fichiers audio :

1. **Chargement du fichier audio** : Le fichier audio est chargé à l'aide de la bibliothèque **torchaudio**, qui permet de traiter les données audio dans un format compatible avec PyTorch.
2. **Rééchantillonnage** : Si le fichier audio ne correspond pas à la fréquence d'échantillonnage attendue par le modèle, une transformation de rééchantillonnage est effectuée pour harmoniser les données.
3. **Conversion stéréo en mono** : Si le fichier audio est stéréo, il est converti en mono en calculant la moyenne des deux canaux.

❖ Fonctionnement du test du modèle

Voici le processus de test utilisé pour tester le modèle AASIST avec un fichier audio donné. Ce processus est détaillé ci-dessous.

1. **Chargement du modèle** :
Le modèle préalablement entraîné est chargé à partir d'un fichier de point de contrôle (checkpoint).
2. **Prétraitement audio** :
Chaque fichier audio est prétraité pour s'assurer qu'il est dans le bon format et qu'il respecte la fréquence d'échantillonnage attendue. Si nécessaire, le fichier est ré échantillonné et converti en mono.
3. **Inférence** :
Une fois l'audio prétraité, il est passé à travers le modèle pour obtenir une prédiction. Le modèle génère une probabilité que le fichier audio soit **genuine** (authentique) ou **spoof** (contrefait).

4. Écriture des résultats dans un fichier log :

Les résultats de la classification sont enregistrés dans un fichier log qui inclut:

- Le nom du fichier audio
- La fréquence d'échantillonnage et la durée
- La décision du modèle (genuine ou spoof)
- Le nom du modèle utilisé pour l'analyse

❖ Résultats :

```
Model loaded successfully.  
Loaded audio: /content/T_0000182355.flac, Sample Rate: 16000  
Model output: tensor([[1.0000e+00, 7.0195e-07, 1.4394e-06, 2.8463e-08, 9.3887e-12, 9.7585e-08,  
8.8163e-12, 3.9147e-09, 3.9146e-09]])  
Log file saved at: /content/output.log
```

Generated Log File Content:

```
Model Name: AASIST  
Metadata:  
  Audio File: /content/T_0000182355.flac  
  Sample Rate: 16000 Hz  
  Duration: 10.01 seconds  
Decision:  
  Bonafide
```

Résultat pour un zip d'audio :

```
Model loaded successfully.  
Processed T_0000000003.flac: spoof  
Processed T_0000000004.flac: spoof  
Processed T_0000000002.flac: spoof  
Processed T_0000000000.flac: spoof  
Processed T_0000000001.flac: spoof  
Processed T_0000182355.flac: bonafide  
Log file saved at: /content/output.log
```

Generated Log File Content:

```
Model Name: AASIST  
  
Audio Processing Results:  
Metadata:  
  Audio File: /content/audio_files/T_0000000003.flac  
  Sample Rate: 16000 Hz  
  Duration: 13.02 seconds  
Decision:  
  Spoof  
  
Metadata:  
  Audio File: /content/audio_files/T_0000000004.flac  
  Sample Rate: 16000 Hz  
  Duration: 15.46 seconds  
Decision:  
  Spoof
```

3. Coding Engineering

a. Création de l'API

L'intégration du modèle AASIST dans une API permet de déployer le système de détection de spoofing audio dans des applications externes et de rendre son utilisation accessible à travers des services web. Cette section détaille la mise en place de l'API, en expliquant les choix techniques, les étapes de développement, et les fonctionnalités disponibles.

b. Choix techniques pour l'API

Le choix de **FastAPI** pour la construction de l'API repose sur ses atouts indéniables en termes de performance et de facilité d'intégration.

En tant que framework moderne, FastAPI est particulièrement adapté aux applications en temps réel grâce à sa gestion optimisée des requêtes HTTP et à sa faible latence. Sa compatibilité avec Python permet une intégration fluide avec des modèles de machine learning développés avec des bibliothèques comme **PyTorch**, simplifiant ainsi l'implémentation des processus d'inférence. De plus, FastAPI se distingue par sa capacité à générer automatiquement une documentation interactive via **Swagger UI**, offrant ainsi une interface intuitive et pratique pour les développeurs afin de tester l'API.

Grâce à ces caractéristiques, l'**API** permet une soumission rapide de fichiers audio et renvoie une prédiction sur leur authenticité sous forme de réponse **JSON**, rendant l'utilisation de l'outil à la fois performante et accessible.

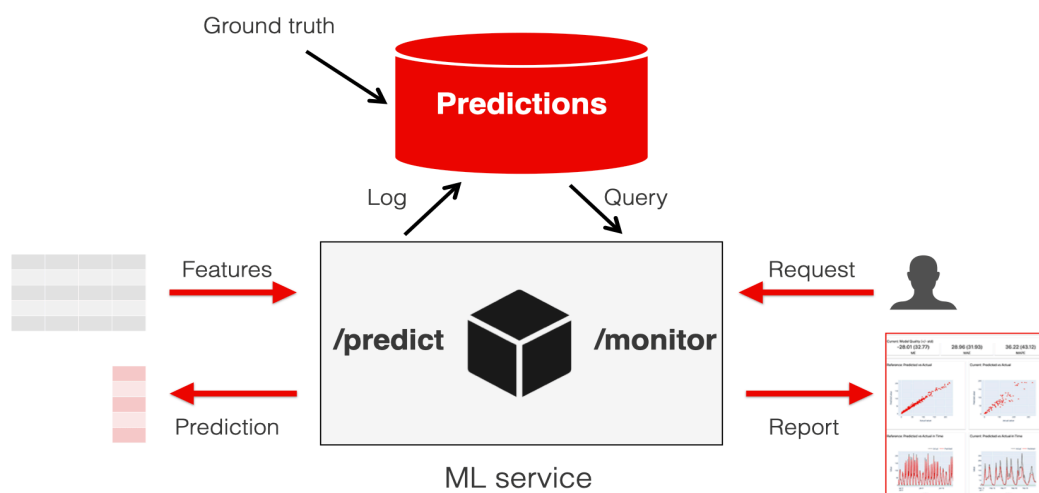


Figure 4 : Architecture API Rest avec ML Service

c. Structure de l'API

L'API suit une architecture simple avec les endpoints suivants :

- **/predict/** : Ce point d'entrée prend un fichier audio en entrée, effectue l'inférence avec le modèle AASIST et retourne un résultat indiquant si le fichier audio est authentique ou spoofé.
 - **Méthode HTTP** : POST
 - **Entrée** : Fichier audio (formats acceptés : .wav, .flac, .mp3)
 - **Sortie** : Résultat de la prédiction sous forme JSON, indiquant la classe prédite (genuine ou spoof), ainsi que la probabilité associée.

d. Développement et Implémentation de l'API

L'implémentation de l'API repose sur les éléments suivants :

Chargement du modèle : À chaque appel de l'API, le modèle AASIST est chargé depuis un fichier de checkpoint pour effectuer l'inférence. Ce processus peut être optimisé en maintenant le modèle en mémoire entre les appels afin de réduire la latence de chargement.

Prétraitement de l'audio : Lorsque l'API reçoit un fichier audio, ce dernier est prétraité pour le convertir en format mono et adapté à la fréquence d'échantillonnage nécessaire. Ce prétraitement est effectué en utilisant **torchaudio** pour garantir que les données audio sont prêtes pour l'inférence.

Prédiction avec le modèle : Une fois l'audio prétraité, il est transmis au modèle pour effectuer la prédiction. Le modèle génère une probabilité d'appartenance à chaque classe (authentique ou spoofé), et le résultat est renvoyé dans la réponse API.

Endpoint API : Endpoint "/predict/" reçoit un fichier audio en entrée, effectue l'inférence et renvoie un résultat sous forme JSON. Ce résultat contient le label prédictif (genuine ou spoof) et un score de confiance, indiquant la probabilité associée à la prédiction.

e. Tester API dans Insomnia

Insomnia est un puissant client REST qui vous permet d'envoyer des requêtes HTTP et d'afficher les réponses des API REST.

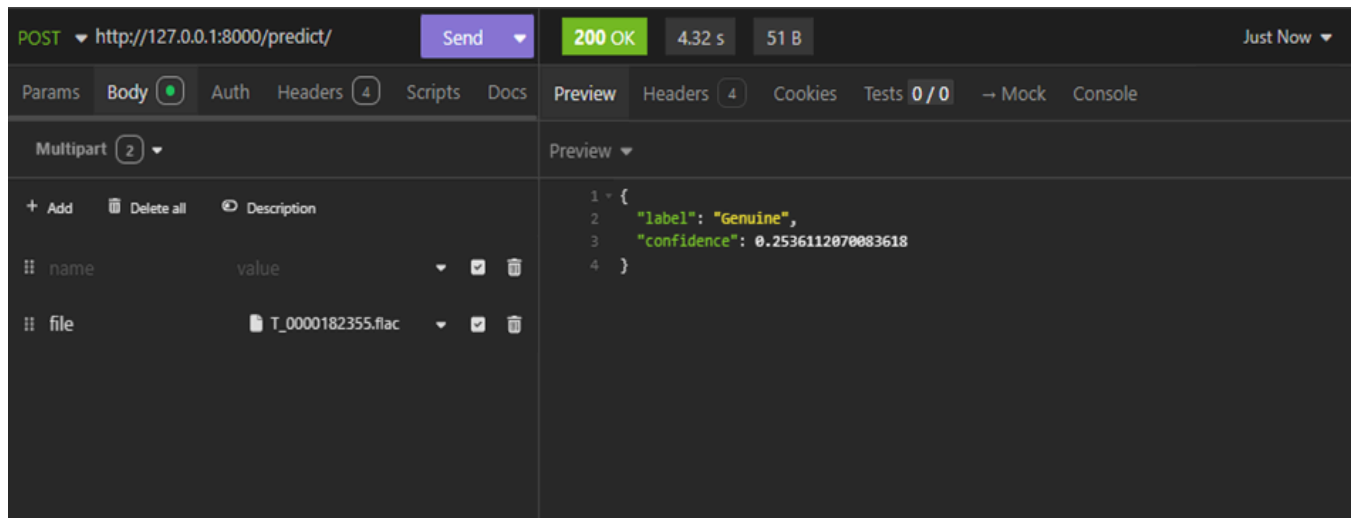


Figure 5 : Réponse de API dans Insomnia

Il est conçu pour être facile à utiliser et aide les développeurs à déboguer et tester leurs API plus efficacement. Avec Insomnia, on peut créer et enregistrer des demandes, les organiser en espaces de travail et les partager avec d'autres.

On peut également créer des environnements personnalisés pour stocker des variables et configurer des paramètres de requête dynamiques, ce qui facilite le test des API qui nécessitent une authentification ou qui doivent envoyer des données différentes pour chaque requête.

4. Conclusion

L'intégration du modèle AASIST dans une API RESTful permet d'offrir une interface flexible et accessible pour la détection de spoofing audio. Grâce à **FastAPI**, l'API est rapide, documentée automatiquement, et facile à utiliser pour les développeurs. En plus de la simplicité d'utilisation, l'API permet des tests et des évaluations rapides des fichiers audio, en offrant des prédictions fiables sur l'authenticité des fichiers audio soumis. Cette intégration offre également une grande scalabilité, rendant le système applicable à des environnements de production à grande échelle.

Chapitre 4 : Conclusion et perspectives

1. Résumé du projet

Ce projet avait pour objectif de détecter les voix frauduleuses et authentiques à l'aide du modèle **AASIST** en développant une interface utilisateur et une API dédiée. Nous avons suivi plusieurs étapes principales :

- Développement d'une interface (frontend) permettant d'enregistrer ou de téléverser un fichier audio.
 - Mise en place d'une API simple pour le traitement des fichiers audio.
 - Intégration et test du modèle **AASIST** pour déterminer si une voix est authentique ou frauduleuse.
- Ces étapes ont permis d'obtenir un système fonctionnel pour analyser des fichiers audio et détecter leur nature.

2. Bilan du travail effectué

a. Points positifs

Le projet a été marqué par plusieurs accomplissements majeurs :

- Développement réussi de l'interface utilisateur pour gérer les fichiers audio.
- Création d'une API efficace pour le traitement et l'analyse des fichiers audio.
- Intégration du modèle AASIST, avec des résultats fiables pour détecter les voix frauduleuses.
- Collaboration et communication efficaces entre les membres de l'équipe, favorisées par une répartition claire des tâches.
- Mise en œuvre de tests pour garantir le bon fonctionnement des différentes composantes.

b. Difficultés rencontrées

Malgré les résultats obtenus, le projet a présenté quelques défis :

- **Optimisation** : Difficultés à traiter des fichiers audio volumineux.
- **Gestion des erreurs** : Problèmes liés aux fichiers mal formatés ou non conformes.

3. Perspectives

a. Court terme

Des actions immédiates peuvent être entreprises pour améliorer le projet :

- Finaliser la **dockerisation de l'API** pour faciliter son déploiement et le rendre portable sur différents environnements.
- Renforcer la gestion des erreurs en prenant en charge davantage de formats audio non conformes et en générant des messages d'erreur plus clairs.

- **Calcul de l'EER (Equal Error Rate)** pour évaluer la performance du modèle AASIST et affiner les paramètres.

b. Moyen terme

Pour étendre le projet à moyen terme, plusieurs objectifs peuvent être définis :

- Ajouter des fonctionnalités avancées à l'API, comme la prise en charge de **détections multi-modèles** pour comparer les performances d'autres algorithmes de détection vocale.
- Développer un **tableau de bord interactif** pour visualiser les résultats des analyses et suivre les logs des fichiers traités.
- Optimiser les performances globales en réduisant les temps de traitement et en minimisant l'utilisation des ressources.

c. Long terme

À plus long terme, le projet pourrait évoluer vers une utilisation dans des environnements réels ou dans le cloud :

- **Déploiement dans le cloud** : Héberger l'API et les services sur une plateforme cloud pour une accessibilité mondiale.
- Ajout d'un module d'**apprentissage continu**, permettant au modèle AASIST de s'améliorer au fil du temps en intégrant de nouvelles données frauduleuses ou authentiques.
- Intégration dans un système plus large de **cybersécurité audio**, combinant la détection de voix frauduleuses avec d'autres méthodes de protection contre les cyberattaques.

Conclusion Générale

Le projet présenté dans ce rapport a permis de répondre à une problématique actuelle et critique dans le domaine de la sécurité des systèmes de reconnaissance vocale : la détection des voix frauduleuses ou falsifiées (spoofing). En combinant les principes de l'intelligence artificielle, du machine learning et des pratiques avancées de MLOps, nous avons développé une solution robuste et évolutive basée sur le modèle **AASIST**.

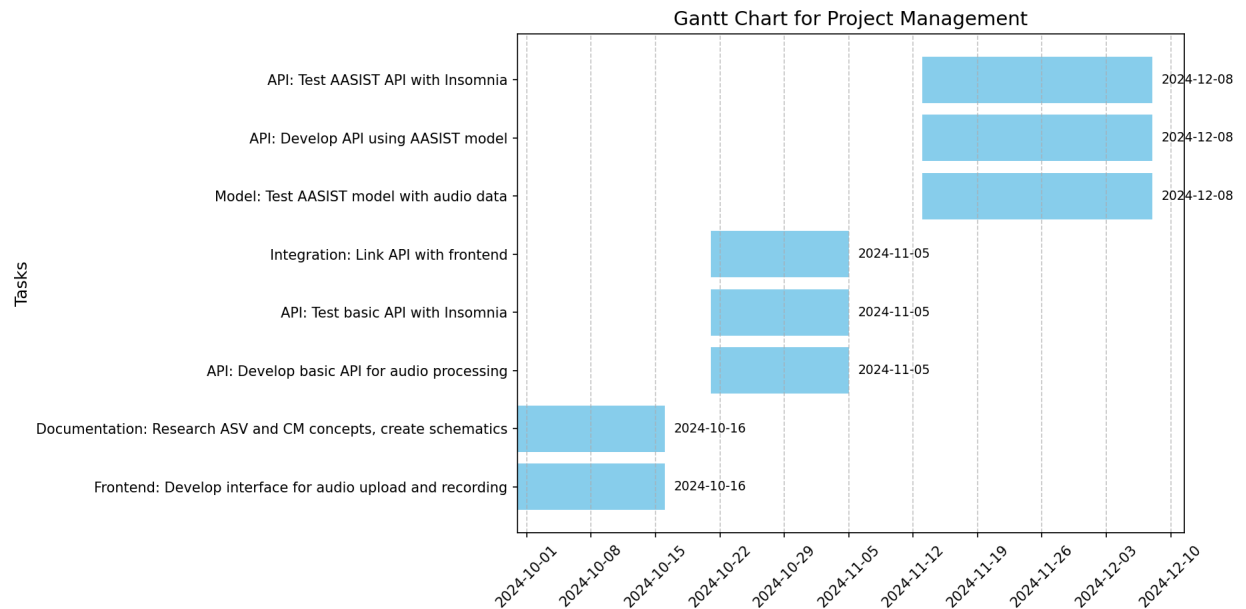
Tout au long du projet, des défis techniques et organisationnels ont été relevés. Grâce à une collaboration efficace entre les membres de l'équipe et à l'utilisation d'outils modernes tels que FastAPI, GitLab, et Insomnia, le projet a atteint ses objectifs principaux.

Les résultats obtenus démontrent la fiabilité et l'efficacité du système développé, tout en mettant en évidence des axes d'amélioration et d'extension. À court terme, des actions telles que la dockerisation de l'API et le renforcement de la gestion des erreurs pourraient être mises en œuvre. À moyen et long terme, le projet pourrait évoluer pour inclure un tableau de bord interactif, l'utilisation de modèles multi-algorithmes, ainsi qu'une intégration dans des environnements cloud pour une accessibilité mondiale.

Ce projet illustre la manière dont les technologies de machine learning, combinées aux bonnes pratiques de MLOps, peuvent relever des défis complexes liés à la sécurité des systèmes audio. Il pose les bases d'une solution évolutive et durable, capable de répondre aux menaces actuelles et futures dans le domaine vocal.

Annexes

Annexe 1



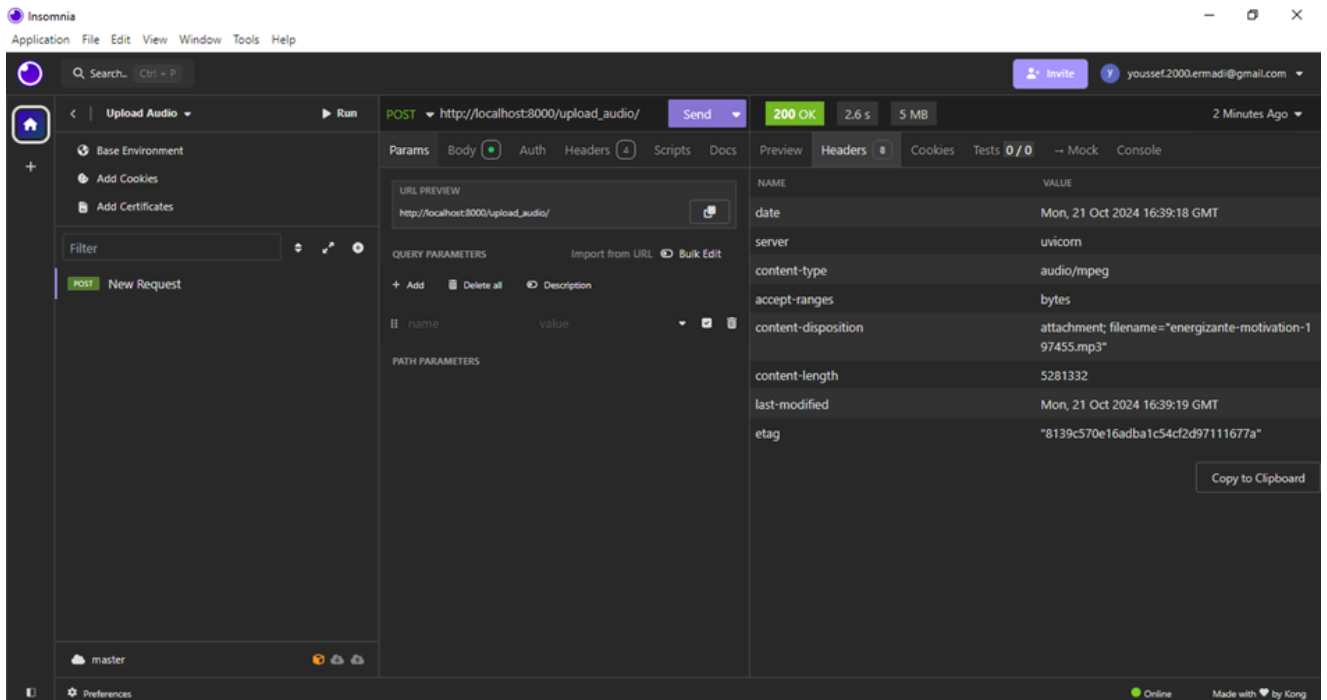
Annexe 2

```

main.py > ...
1  from fastapi import FastAPI, UploadFile, File
2  from fastapi.responses import FileResponse
3  import os
4  import shutil
5  import mimetypes
6
7  app = FastAPI()
8
9  TEMP_DIR = "temp_audio"
10 os.makedirs(TEMP_DIR, exist_ok=True)
11
12 @app.post("/upload_audio/")
13 async def upload_audio(file: UploadFile = File(...)):
14     try:
15         # Chemin du fichier temporaire
16         temp_file_path = os.path.join(TEMP_DIR, file.filename)
17
18         # Sauvegarder le fichier audio tel quel sans le traiter
19         with open(temp_file_path, "wb") as buffer:
20             shutil.copyfileobj(file.file, buffer)
21
22         # Obtenir le type MIME correct en fonction de l'extension du fichier
23         mime_type, _ = mimetypes.guess_type(temp_file_path)
24
25         # Retourner le fichier avec le bon Content-Type
26         return FileResponse(temp_file_path, media_type=mime_type, filename=file.filename)
27     except Exception as e:
28         Loading...
29         return {"error": str(e)}
30

```

```
PS C:\Users\Pc\Desktop\cours\S9\Projet5A> uvicorn main:app --reload
?[32mINFO?[0m:      Will watch for changes in these directories: ['C:\\Users\\Pc\\Desktop\\cours\\S9\\Projet5A']
?[32mINFO?[0m:      Uvicorn running on ?[1mhttp://127.0.0.1:8000?[0m (Press CTRL+C to quit)
?[32mINFO?[0m:      Started reloader process [?[36m?[1m19404?[0m] using [?[36m?[1mStatReload?[0m]
?[32mINFO?[0m:      Started server process [?[36m5092?[0m]
?[32mINFO?[0m:      Waiting for application startup.
?[32mINFO?[0m:      Application startup complete.
?[32mINFO?[0m:      127.0.0.1:56871 - "[?[1mPOST /upload_audio/ HTTP/1.1?[0m" ?[32m200 OK?[0m
```



The screenshot shows the Insomnia application interface. The top bar indicates the user is logged in as 'youssef.2000.ermadi@gmail.com'. The main panel displays a REST client view for a POST request to 'http://localhost:8000/upload_audio/'. The request status is '200 OK' with a response time of '2.6 s' and a body size of '5 MB'. The response headers are visible, including 'date', 'server', 'content-type', 'accept-ranges', 'content-disposition', 'content-length', 'last-modified', and 'etag'. The 'Content-Disposition' header shows the filename 'energizante-motivation-197455.mp3'.

```
# Configurer le middleware CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"], # Remplacez "*" par l'origine spécifique si nécessaire
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Références :

- [1] "What Is Insomnia, the API REST Client? – Assistance & Documentation – ProAbono." *Assistance & Documentation – ProAbono – Find All Questions and Answers*, <https://docs.proabono.com/documentation/api-overview/what-is-insomnia-the-api-rest-client/>. Accessed 8 Dec. 2024.
- [2] "Databases | ASVspoof – Dataset" <https://www.asvspoof.org/database>
- [3] <https://e-gitlab.univ-lemans.fr/mshamsi/ASVSpooF5>
- [4] clovaai/aasist: Official PyTorch implementation of "AASIST: Audio Anti-Spoofing using Integrated Spectro-Temporal Graph Attention Networks"
<https://github.com/clovaai/aasist>
- [5] AASIST: Audio Anti-Spoofing using Integrated Spectro-Temporal Graph Attention Networks Jung et al. <https://arxiv.org/abs/2110.01200>
- [6] (PDF) AASIST: Audio Anti-Spoofing Using Integrated Spectro-Temporal Graph Attention Networks,
www.researchgate.net/publication/360792683_AASIST_Audio_Anti-Spoofing_Using_Integrated_Spectro-Temporal_Graph_Attention_Networks. Accessed 8 Dec. 2024.
- [7] "AASIST: Audio Anti-Spoofing Using Integrated Spectro-Temporal Graph Attention Networks." Ar5iv, ar5iv.labs.arxiv.org/html/2110.01200. Accessed 9 Dec. 2024.