Liste des Sujets de PFA en Machine Learning et Deep Learning

Pr. Soufiane Hamida

12 mars 2025

Introduction

Ce document présente une liste de sujets de projet de fin d'année axés sur le Machine Learning (ML) et le Deep Learning (DL). Ces sujets sont conçus pour être réalisables en deux mois tout en restant innovants. Chaque sujet propose des technologies variées pour le backend et le frontend, ainsi qu'une planification détaillée.

1 Sujets de Projet

1.1 Système de Recommandation de Films basé sur le Machine Learning

Description : Développer un système de recommandation de films qui utilise des algorithmes de Machine Learning pour suggérer des films aux utilisateurs en fonction de leurs préférences et de leur historique de visionnage.

Objectifs :

- Collecter et prétraiter des données de films (par exemple, via des APIs comme TMDB).
- Implémenter un algorithme de recommandation (filtrage collaboratif ou basé sur le contenu).
- Créer une interface utilisateur simple pour afficher les recommandations.

Technologies :

- **ML/DL :** Scikit-learn, TensorFlow, ou PyTorch.
- **Backend :** Flask ou Django.
- **Frontend :** React.js ou Vue.js.
- **Base de Données :** SQLite ou MongoDB.

Planification :

- Semaine 1 : Collecte et prétraitement des données.
- Semaine 2-3 : Développement de l'algorithme de recommandation.
- Semaine 4-5 : Création de l'interface utilisateur.
- Semaine 6-7 : Tests et documentation.

1.2 Classification d'Images avec Deep Learning (MNIST ou CIFAR-10)

Description : Créer un modèle de Deep Learning pour classer des images à partir de jeux de données populaires comme MNIST (chiffres manuscrits) ou CIFAR-10 (objets divers).

Objectifs :

- Préparer et prétraiter les données d'images.
- Entraîner un modèle de classification avec un réseau de neurones convolutifs (CNN).
- Évaluer la performance du modèle et afficher les résultats dans une interface simple.

Technologies :

- **DL :** TensorFlow, Keras, ou PyTorch.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.
- **Base de Données :** Non nécessaire (utilisation de fichiers locaux).

Planification :

- Semaine 1 : Collecte et prétraitement des données.
- Semaine 2-3 : Entraînement du modèle CNN.
- Semaine 4-5 : Création d'une interface pour afficher les résultats.
- Semaine 6-7 : Tests et documentation.

1.3 Analyse de Sentiments dans les Tweets avec NLP

Description : Développer un modèle de traitement du langage naturel (NLP) pour analyser les sentiments (positif, négatif, neutre) dans des tweets ou des commentaires en ligne.

Objectifs :

- Collecter des données textuelles (via l'API Twitter ou des datasets publics).
- Prétraiter les textes (tokenisation, suppression des stopwords, etc.).
- Entraîner un modèle de classification de sentiments avec des techniques de NLP.

Technologies :

- **NLP :** Hugging Face Transformers, SpaCy, ou NLTK.
- **ML/DL :** TensorFlow, PyTorch, ou Scikit-learn.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.

Planification :

- Semaine 1 : Collecte et prétraitement des données textuelles.
- Semaine 2-3 : Entraînement du modèle de classification de sentiments.
- Semaine 4-5 : Création d'une interface pour afficher les résultats.
- Semaine 6-7 : Tests et documentation.

1.4 Prédiction des Prix des Logements avec Machine Learning

Description : Créer un modèle de Machine Learning pour prédire les prix des logements en fonction de caractéristiques telles que la superficie, le nombre de chambres, la localisation, etc.

Objectifs :

- Collecter et prétraiter des données sur les logements (par exemple, via des datasets publics comme Kaggle).
- Entraîner un modèle de régression (Random Forest, Gradient Boosting, ou Réseaux de Neurones).
- Afficher les prédictions dans une interface utilisateur.

Technologies :

- **ML :** Scikit-learn, XGBoost, ou TensorFlow.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.
- **Base de Données :** SQLite ou PostgreSQL.

Planification :

- Semaine 1 : Collecte et prétraitement des données.
- Semaine 2-3 : Entraînement du modèle de régression.
- Semaine 4-5 : Création d'une interface pour afficher les prédictions.
- Semaine 6-7 : Tests et documentation.

1.5 Détection de Fraude Bancaire avec Machine Learning

Description : Développer un modèle de Machine Learning pour détecter les transactions frauduleuses dans un jeu de données de transactions bancaires.

Objectifs :

- Collecter et prétraiter des données de transactions bancaires (par exemple, via des datasets publics).
- Entraîner un modèle de classification (Logistic Regression, Random Forest, ou Réseaux de Neurones).
- Évaluer la performance du modèle et afficher les résultats dans une interface.

Technologies :

- **ML :** Scikit-learn, TensorFlow, ou PyTorch.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.
- **Base de Données :** SQLite ou PostgreSQL.

Planification :

- Semaine 1 : Collecte et prétraitement des données.
- Semaine 2-3 : Entraînement du modèle de classification.
- Semaine 4-5 : Création d'une interface pour afficher les résultats.
- Semaine 6-7 : Tests et documentation.

1.6 Génération de Texte avec des Modèles de Langage (GPT-like)

Description : Utiliser un modèle de langage pré-entraîné (comme GPT-2 ou GPT-3) pour générer du texte automatiquement en fonction d'une entrée utilisateur.

Objectifs :

- Intégrer un modèle de langage pré-entraîné (par exemple, via Hugging Face Transformers).
- Créer une interface utilisateur pour permettre à l'utilisateur de saisir du texte et de recevoir une réponse générée.
- Évaluer la qualité du texte généré.

Technologies :

- **NLP :** Hugging Face Transformers, TensorFlow, ou PyTorch.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.
- **Base de Données :** Non nécessaire.

Planification :

- Semaine 1 : Intégration du modèle de langage pré-entraîné.
- Semaine 2-3 : Création de l'interface utilisateur.
- Semaine 4-5 : Tests et évaluation de la qualité du texte généré.
- Semaine 6-7 : Documentation et présentation.

1.7 Reconnaissance d'Objets en Temps Réel avec YOLO

Description : Développer une application de reconnaissance d'objets en temps réel en utilisant le modèle YOLO (You Only Look Once) pour détecter des objets dans une vidéo ou une image.

Objectifs :

- Intégrer un modèle YOLO pré-entraîné (par exemple, YOLOv5).
- Créer une interface utilisateur pour afficher les résultats de la détection en temps réel.
- Tester le modèle sur des vidéos ou des images.

Technologies :

- **DL :** PyTorch, OpenCV, YOLOv5.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.
- **Base de Données :** Non nécessaire.

Planification :

- Semaine 1 : Intégration du modèle YOLO.
- Semaine 2-3 : Création de l'interface utilisateur.
- Semaine 4-5 : Tests en temps réel avec des vidéos ou des images.
- Semaine 6-7 : Documentation et présentation.

1.8 Prédiction de la Qualité de l'Air avec Machine Learning

Description : Créer un modèle de Machine Learning pour prédire la qualité de l'air en fonction de données météorologiques et de capteurs.

Objectifs :

- Collecter et prétraiter des données sur la qualité de l'air (par exemple, via des datasets publics).
- Entraîner un modèle de régression ou de classification.
- Afficher les prédictions dans une interface utilisateur.

Technologies :

- **ML :** Scikit-learn, TensorFlow, ou PyTorch.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.
- **Base de Données :** SQLite ou PostgreSQL.

Planification :

- Semaine 1 : Collecte et prétraitement des données.
- Semaine 2-3 : Entraînement du modèle.
- Semaine 4-5 : Création de l'interface utilisateur.
- Semaine 6-7 : Tests et documentation.

1.9 Reconnaissance Vocale avec Deep Learning

Description : Développer un système de reconnaissance vocale qui convertit la parole en texte en utilisant des modèles de Deep Learning.

Objectifs :

- Intégrer un modèle de reconnaissance vocale pré-entraîné (par exemple, via Hugging Face ou TensorFlow).
- Créer une interface utilisateur pour enregistrer et transcrire la parole.
- Évaluer la précision du modèle.

Technologies :

- **DL :** TensorFlow, PyTorch, ou Hugging Face Transformers.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.
- **Base de Données :** Non nécessaire.

Planification :

- Semaine 1 : Intégration du modèle de reconnaissance vocale.
- Semaine 2-3 : Création de l'interface utilisateur.
- Semaine 4-5 : Tests et évaluation de la précision.
- Semaine 6-7 : Documentation et présentation.

1.10 Détection de Fausses Nouvelles avec NLP

Description : Créer un modèle de Machine Learning pour détecter les fausses nouvelles (fake news) en analysant le contenu textuel des articles.

Objectifs :

- Collecter et prétraiter des données textuelles (par exemple, via des datasets publics).
- Entraîner un modèle de classification pour distinguer les vraies des fausses nouvelles.
- Afficher les résultats dans une interface utilisateur.

Technologies :

- **NLP :** Hugging Face Transformers, SpaCy, ou NLTK.
- **ML/DL :** TensorFlow, PyTorch, ou Scikit-learn.
- **Backend :** Flask ou FastAPI.
- **Frontend :** Streamlit ou React.js.

Planification :

- Semaine 1 : Collecte et prétraitement des données.
- Semaine 2-3 : Entraînement du modèle de classification.
- Semaine 4-5 : Création de l'interface utilisateur.
- Semaine 6-7 : Tests et documentation.