Ecole Publique d'Ingénieurs en 3 ans

Rapport

DETECTION DE PROPAGANDE TERRORISTE

Projet 2A

AMODE MALL Samir, BRISSOT Simon, CASTETZ Mathieu, SILAIRE Igor, ZYGMUNTOWICZ Patrick
Année Universitaire 2024/2025

2A Informatique, EPCS, CYA

Tuteur ENSICAEN : **Tanguy GERNOT, Emmanuel GIGUET**

TABLE DES MATIERES

Remerciements

Nous remercions nos encadrants M. Tanguy Gernot et M. Emmanuel Giguet pour leur accompagnement, leurs conseils précieux et leurs expertises, qui nous ont guidés tout au long de ce travail. Leurs soutiens et leurs retours constructifs nous ont permis d'améliorer notre méthodologie et d'affiner nos analyses.

1. Contexte et problématique

1.1. Introduction

L'essor du numérique et des plateformes en ligne a facilité la diffusion d'informations à grande échelle, mais a également permis la propagation de contenus problématiques, notamment liés à la propagande terroriste. Face à cette menace, notre projet vise à développer un outil capable d'identifier ces contenus en combinant différentes approches d'analyse.

Initialement, nous avons opté pour une approche basée sur un filtrage par mots-clés, permettant une première classification des documents textuels. Cependant, cette méthode s'est rapidement heurtée à des limites, notamment en raison du manque d'accès aux listes de mots sensibles utilisées par les organismes spécialisés. Afin de pallier ces contraintes et d'améliorer la précision de l'analyse, nous avons intégré un modèle de traitement du langage naturel (LLM) via Hugging Face.

Le projet a ensuite été élargi à l'analyse d'images et d'audios, permettant une identification plus complète des contenus suspects. Enfin, pour répondre aux exigences des différents utilisateurs (agences gouvernementales, modérateurs de plateformes, etc.), nous avons rendu le modèle personnalisable, afin que chaque organisme puisse ajouter ses propres filtres et affiner la détection.

Ce rapport détaille les étapes de conception et de développement du projet, ainsi que les défis rencontrés et les perspectives d'amélioration futures.

1.2. Définition du problème

La prolifération de contenus à caractère terroriste sur les plateformes numériques constitue un enjeu majeur pour les autorités et les plateformes de modération. Ces contenus peuvent inclure des discours de propagande, des appels à la violence, des symboles spécifiques ou encore des contenus multimédias codés. Leur détection est cruciale pour prévenir la radicalisation en ligne et lutter contre la diffusion d'idéologies extrémistes.

1.3. Enjeux liés à la détection de contenus terroristes

Les principaux enjeux liés à la détection de ces contenus sont les suivants :

1. Précision et fiabilité : Réduire les faux positifs et faux négatifs pour éviter la suppression abusive de contenus légitimes ou la non-détection de contenus problématiques.

2. Évolutivité : Adapter les modèles de détection aux nouvelles stratégies employées par les groupes extrémistes.
3. Multimodalité : Étendre la détection au-delà du texte, en intégrant l'analyse d'images et d'audio.
4. Personnalisation : Permettre aux organismes d'affiner les filtres et les critères de détection selon leurs besoins spécifiques.
5. Sécurisation des données : Protéger les informations sensibles et éviter toute fuite de données.

1.4. Informations initiales avant le brainstorming

Avant d'entamer nos discussions, nous disposions des informations suivantes concernant la problématique :

- **Identification des contenus liés à la propagande terroriste** :
  - Canaux de communication concernés : audio, vidéo, textuel, réseaux sociaux, forums, blogs, jeux vidéo, etc.
  - Nature des contenus analysables : photos, vidéos, textes extraits d'images (OCR), commentaires, publications sur les réseaux, fichiers audios.
  - Outils possibles : Speech-To-Text (STT) et traduction automatique pour les vidéos, OCR pour les images contenant du texte.
  - Priorité aux outils fonctionnant en local, bien que des systèmes en ligne puissent être testés ponctuellement.
  - Recherche de bases de données existantes et de méthodes issues d'articles scientifiques ou de compétitions Kaggle.

Les enquêteurs étant susceptibles de travailler avec des contenus dans des langues qu'ils ne maîtrisent pas, il était également important de prévoir une stratégie pour déterminer rapidement si un document est pertinent ou non.

1.5. Résultats du brainstorming

Lors de notre première journée de travail, nous avons mené une réflexion approfondie pour structurer le projet.

Personas identifiés

Nous avons défini plusieurs types d'utilisateurs potentiels pour mieux cerner les besoins :

Tableau 1 : Personnas

Pseudo	Age	Profil	Attentes
Camille	52 ans	Cadre justice	Informations précises, impossibilité d'avoir des faux positifs, production de rapports complets
Modo Discord	20 ans	Modérateur	Filtrer en continu, classer les posts suspects par priorité
OSS 117	38 ans	DGSE	Notifier en cas de nouvelles informations, suivi de profils spécifiques

Nous avons décidé de nous concentrer sur le persona **OSS 117** pour cette phase, car ses besoins sont les plus pertinents et réalisables dans les délais impartis.

Architecture logicielle

Le logiciel sera composé de trois grandes parties :

1. **Configuration** : Choix des paramètres de recherche (sources, filtres, etc.).
2. **Analyse des données** : Recherches automatisées sur les bases de données et plateformes.
3. **Production de rapports** : Résultats exploitables sous forme de rapport.

Proposition de MVP (Minimum Viable Product)

Tableau 2 : MVP

Fonctionnalités	Config	Analyse	Rapport
MUST	Choisir la BDD à analyser, définir les filtres	Scanner automatiquement forums, réseaux sociaux, sites web	Génération de rapports formatés
COULD	Intégrer des bases de données personnalisées	Analyser des discours subliminaux, identifier les niveaux de menace	Priorisation des alertes
SHOULD	Surveillance des réseaux clandestins et darkweb	Analyse des liens entre groupes de propagande	Détection du ton émotionnel

Nous avons décidé de nous concentrer sur **l'interprétation de texte** en trois étapes :

1. Identifier les mots problématiques.

2. Analyser la phrase contenant ces mots.
3. Évaluer le contexte global pour prendre une décision.

2. Objectifs du projet

2.1. Objectifs généraux

L'objectif principal du projet est de concevoir une solution logicielle permettant d'identifier et d'analyser des contenus potentiellement liés à la propagande terroriste. Cette solution doit être :

- Efficace : Capable de traiter de grands volumes de données avec une précision élevée.
- Évolutive : Adaptable aux évolutions des discours et des techniques de contournement.
- Sécurisée : Assurant la confidentialité des données analysées.
- Personnalisable : Permettant aux utilisateurs d'intégrer leurs propres filtres et modèles.

2.2. Objectifs spécifiques

1. Développer un premier système de filtrage basé sur une liste de mots-clés sensibles.
2. Intégrer un modèle de traitement du langage naturel (LLM) pour une analyse contextuelle plus fine.
3. Étendre l'analyse aux contenus multimédias (images et audio).
4. Permettre aux utilisateurs d'ajouter leurs propres données pour affiner les analyses.
5. Assurer une gestion sécurisée des accès et des modèles via une interface utilisateur dédiée.

3. Méthodologie et approche

3.1. Première approche : Filtrage par mots-clés

Dans un premier temps, une méthode basée sur le filtrage par mots-clés a été implémentée pour identifier les documents suspects. Cette approche repose sur l'utilisation de la

bibliothèque **spaCy** et de son module **PhraseMatcher**, qui permet de repérer des expressions prédéfinies dans un texte.

3.1.1. Fonctionnement de l'algorithme

1. **Initialisation du modèle linguistique** :
    a. Utilisation du modèle **fr_core_news_sm** de spaCy pour l'analyse de texte en français.
2. **Définition d'une liste de mots-clés** :
    a. Une liste contenant des termes potentiellement liés à la propagande terroriste (ex. : "djihad", "kalachnikov", "attentat").
    b. Ces mots sont convertis en objets spaCy utilisables par le matcher.
3. **Analyse de documents** :
    a. Le programme charge chaque document texte dans un dossier source.
    b. Chaque document est analysé par le modèle spaCy.
    c. Si un ou plusieurs mots-clés sont détectés, le fichier est identifié comme suspect.
4. **Tri et stockage des fichiers suspects** :
    a. Les fichiers contenant des mots-clés sont déplacés vers un dossier "suspects" pour une analyse plus approfondie.

3.1.2. Avantages et limites de cette approche

Avantages :

- Implémentation simple et rapide.
- Bonne capacité à filtrer un premier niveau de contenu.

Limites :

Cette approche présente plusieurs limites :

- **Faux Positifs :** Fort taux de faux positifs (certains mots peuvent être utilisés hors contexte terroriste).
- **Faux négatifs :** Un texte peut relever de la propagande sans pour autant contenir de mots contenus dans la liste.
- **Temps de traitement** : Lors de l'analyse de grands volumes de textes, le programme devient lent et peu performant.
- **Absence de contexte** : Un mot-clé seul ne permet pas toujours de juger si un texte est réellement problématique.
- **Sensibilité aux variations linguistiques** : Une reformulation ou l'utilisation de synonymes peut contourner le filtre.

3.1.3. Ajustements et intégration dans le projet

Face à ces limites, nous avons initialement envisagé d'utiliser ce filtre comme un **pré-traitement avant l'analyse par un LLM**. L'idée était de réduire le volume de texte à analyser par l'IA pour améliorer les performances. Cependant, après plusieurs tests, nous avons constaté que ce **n'était pas nécessaire**, car le modèle LLM est capable de traiter directement les textes de manière efficace.

Finalement, nous avons décidé de **conserver ce programme dans le projet** en tant que module complémentaire, mais **il n'a pas été intégré dans la version finale du logiciel**.

3.2. Amélioration avec un modèle LLM

Face aux limitations du simple filtrage par mots-clés, nous avons intégré un modèle de type **LLM (Large Language Model)** capable d'analyser le contexte des textes et d'en extraire une signification plus précise.

3.2.1. Comparaison des modèles de langage

Avant de sélectionner notre modèle, nous avons comparé les approches **MLM (Masked Language Model)** et **LLM (Large Language Model)** selon plusieurs critères :

Tableau 3 : Comparaison MLM et LLM

Critères	MLM	LLM
But	Analyse + compréhension + extraction de texte	Compréhension + génération de texte
Performance	Très performant après un fine-tuning	Très performant mais risque de surdimensionnement
Coût de calcul	Faible, rapide et peu de ressources nécessaires	Élevé, nécessite beaucoup de ressources et de mémoire
Complexité	Simple, modules légers et peu complexes	Modules plus lourds et coûteux

3.2.2. Choix du modèle

Le modèle utilisé est **"HomardBacon/distilbert-propagande-detector"**, un modèle basé sur DistilBERT, entraîné spécifiquement par nous pour classifier les contenus en plusieurs catégories :

- **Propagande**
- **Neutre**
- **Opinion**
- **Fait**
- **Autre**

3.2.3. Fonctionnement du pipeline d'analyse

L'analyse est réalisée en plusieurs étapes :

1. **Tokenisation et passage dans le modèle** : le texte est transformé en tokens et analysé par le modèle pour attribuer une classe.
2. **Interprétation des résultats** : le modèle fournit un score de confiance pour la classification.
3. **Explication des décisions** : grâce à **transformers_interpret**, nous identifions les mots les plus influents dans la classification.

3.2.4. Avantages de cette approche

- **Meilleure prise en compte du contexte**, réduisant les faux positifs.
- **Analyse des mots influents**, permettant d'expliquer les décisions du modèle.
- **Adaptabilité**, le modèle peut être affiné avec de nouvelles données.

3.3. Extension à l'analyse multimédia

3.3.1. Traitement de l'audio

Dans le cadre de l'extension de notre projet à l'analyse multimédia, nous avons intégré un module de traitement de l'audio basé sur le modèle **Whisper** d'OpenAI. Ce module permet la transcription automatique des fichiers audio en texte, facilitant ainsi l'analyse des discours potentiellement problématiques.

Fonctionnement du module audio

1. **Transcription de l'audio** : Le fichier audio est analysé par Whisper, qui génère une transcription textuelle.

2. **Analyse textuelle** : Le texte obtenu est ensuite soumis à notre pipeline d'analyse basé sur le LLM, permettant de détecter d'éventuels contenus suspects.
3. **Retour des résultats** : Le système fournit un rapport indiquant les segments de l'audio considérés comme problématiques.

Avantages de cette approche

- **Précision élevée** : Whisper est capable de traiter divers accents et bruits de fond.
- **Automatisation** : Permet une analyse rapide et efficace sans intervention humaine.
- **Complémentarité** : L'intégration avec notre modèle LLM améliore la compréhension contextuelle des contenus transcrits

3.3.2. Traitement des images

Toujours dans l'idée d'étendre les possibilités de l'analyse, nous avons intégré plusieurs modules de reconnaissance automatique de caractères (OCR) tels que Tesseract, PaddleOCR et EasyOCR. Le but ici est d'extraire le texte contenu dans des images afin d'ensuite pouvoir l'analyser.

Fonctionnement du module OCR

1. **Prétraitement de l'image :** L'image est chargée par la classe ImageAnalyzer de **pictures_analysis.py** puis améliorée par **preprocessing.py** afin d'optimiser la reconnaissance du texte.
2. **Extraction du texte :** Celle-ci se réalise avec 3 moteurs OCR dans **ocr.py**. Il s'agit de **Tesseract** (modèle classique), **PaddleOCR** (optimisé pour la détection multilingue) et **EasyOCR** (utilise des réseaux de neurones pour améliorer la reconnaissance). Chacun de ses moteurs extrait indépendamment le texte de l'image.
3. **Fusion des résultats :** Le fichier **fusion.py** s'occupe de comparer et fusionner les différentes transcriptions. L'algorithme, basé sur **fuzzywuzzy**, sélectionne la version de chaque ligne qu'il estime être la plus fiable.
4. **Correction :** Le texte obtenu est traité dans **correction.py**. Un correcteur basé sur **SpellChecker** ajuste les potentielles erreurs.
5. **Retour des résultats** : Le texte final est retourné par la fonction **analyze_image** de **ImageAnalyzer**. Le texte obtenu peut ensuite être analysé de la même manière que l'audio ou le texte brut via le pipeline de détection.

Avantages de cette approche

- **Robustesse :** L'utilisation de plusieurs moteurs OCR aux forces diverses permet d'améliorer la précision de l'extraction, en ne choisissant pas une des analyses mais bien en sélectionnant dynamiquement la meilleure version de chaque ligne via fuzzywuzzy.
- **Correction automatique :** Les fautes dues à une mauvaise reconnaissance sont corrigées avant de passer à l'analyse.
- **Complémentarité** : L'intégration avec le modèle LLM étend l'identification de contenus problématiques aux images

4. Architecture du système

4.1. Système d'authentification et gestion des accès

Afin d'assurer la sécurité de l'accès à notre application, nous avons mis en place un système d'authentification basé sur un mécanisme de **hashing sécurisé des mots de passe** et des **tokens JWT (JSON Web Token)** pour la gestion des sessions.

Fonctionnement du système

L'authentification repose sur plusieurs étapes :

1. **Stockage sécurisé des mots de passe** : Utilisation de **bcrypt** pour le hachage avant stockage.
2. **Vérification des identifiants** : Comparaison entre le mot de passe saisi et son hachage enregistré.
3. **Génération de tokens JWT** : Un token avec expiration de **30 minutes** est créé pour authentifier les requêtes futures.

4.2. Workflow général

Le processus de détection suit un pipeline structuré :

1. Collecte des données (texte, image, audio).
2. Prétraitement des données.

3. Analyse par les modèles spécifiques à chaque modalité.
4. Fusion et classification des résultats.
5. Génération d'un rapport détaillé pour l'utilisateur.

5. Développement et mise en œuvre

5.1. Génération de rapports

L'un des modules clés du projet est la génération automatique de rapports détaillés à partir des analyses effectuées. Pour cela, nous avons développé deux scripts principaux :

- **pdf_generator.py** : Ce module utilise la bibliothèque reportlab pour créer des rapports PDF structurés avec les résultats de l'analyse. Il permet d'ajouter des titres, du texte mis en forme, ainsi que des graphiques pour illustrer les résultats.
- **report_generator.py** : Ce script récupère les données issues des analyses textuelles, d'images et d'audios, puis les formate avant de les transmettre à pdf_generator.py. Il inclut également une fonction de surlignage des mots-clés détectés pour une meilleure visualisation des éléments critiques.

L'objectif de ces modules est de fournir aux utilisateurs un moyen rapide et efficace d'obtenir un compte-rendu clair des analyses réalisées, facilitant ainsi la prise de décision et le suivi des menaces détectées.

5.2. Interface graphique et expérience utilisateur

L'interface graphique (UI) du projet a été développée en **PyQt5**, offrant une expérience utilisateur fluide et intuitive. Elle est structurée en plusieurs fenêtres permettant d'accéder aux différentes fonctionnalités d'analyse et de gestion des résultats.

Composants principaux de l'UI

- **Fenêtre principale (main_window.py)** : Point d'entrée de l'application, permettant de naviguer entre les différentes analyses.
- **Fenêtre d'analyse (analysis_window.py)** : Permet de charger du texte, des images ou des fichiers audios pour lancer l'analyse.
- **Fenêtre des résultats (results_window.py)** : Affiche les résultats avancés de l'analyse, avec des détails sur la classification et les scores de confiance.

- **Fenêtre d'authentification (login_window.py)** : Assure la gestion des accès à l'application via un système sécurisé basé sur JWT.
- **Gestion du style (style.qss)** : Fournit un design cohérent à l'application via un fichier de style dédié.

Fonctionnalités clés

- **Importation de fichiers** : L'utilisateur peut charger un fichier texte, image ou audio.
- **Analyse en temps réel** : Un thread dédié assure que l'interface reste réactive pendant l'exécution des analyses.
- **Génération de rapports PDF** : Les résultats d'analyse peuvent être exportés sous forme de document PDF.
- **Navigation fluide** : Les fenêtres sont interconnectées pour assurer une utilisation intuitive.

L'intégration d'une interface graphique claire et ergonomique a permis d'améliorer l'accessibilité du projet, rendant son utilisation plus intuitive pour les modérateurs et analystes.

5.3. Entraînement du Modèle

L'entraînement du modèle repose sur une approche de distillation des connaissances, où un modèle léger (DistilBERT) apprend à partir d'un modèle plus lourd (BERT). Cette approche permet d'obtenir un modèle performant avec un temps d'inférence réduit.

5.3.1. Chargement des données

Le script train_model.py charge le dataset en utilisant datasets.load_dataset, qui récupère les données à partir du fichier CSV. Le dataset est ensuite tokenisé en appliquant la fonction de tokenisation définie dans preprocessing.py.

5.3.2. Modèles utilisés

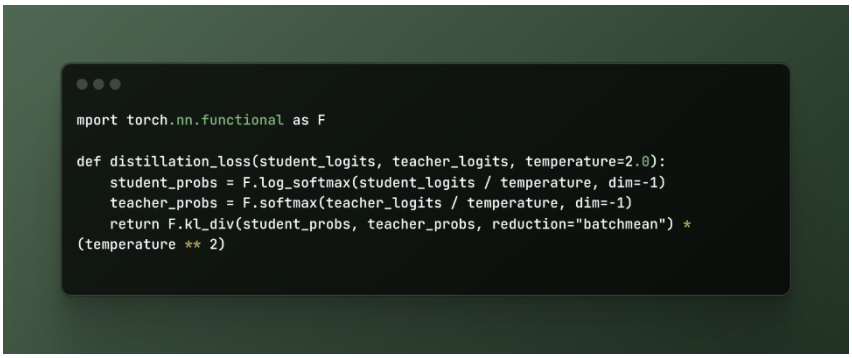
Deux modèles sont impliqués :

Modèle enseignant (teacher model) : Un modèle BERT pré-entraîné pour la classification de propagande, chargé depuis ./models/bert_propagande_detector.

Modèle étudiant (student model) : Un modèle DistilBERT entraîné à partir des prédictions du modèle enseignant.

5.3.3. Perte de distillation

Une fonction de perte spécifique est utilisée pour la distillation des connaissances :



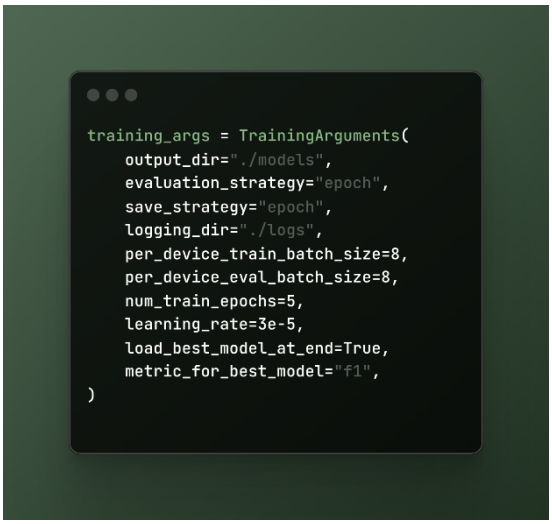
```
mport torch.nn.functional as F

def distillation_loss(student_logits, teacher_logits, temperature=2.0):
    student_probs = F.log_softmax(student_logits / temperature, dim=-1)
    teacher_probs = F.softmax(teacher_logits / temperature, dim=-1)
    return F.kl_div(student_probs, teacher_probs, reduction="batchmean") *
(temperature ** 2)
```

Figure 1 : Fonction distillation

Cette fonction applique une divergence de Kullback-Leibler (KL-Divergence) pour aligner les prédictions du modèle étudiant sur celles du modèle enseignant.

5.3.4. Entraînement

L'entraînement est effectué à l'aide de la classe Trainer de transformers, avec les paramètres suivants :



```
training_args = TrainingArguments(
    output_dir="./models",
    evaluation_strategy="epoch",
    save_strategy="epoch",
    logging_dir="./logs",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=5,
    learning_rate=3e-5,
    load_best_model_at_end=True,
    metric_for_best_model="f1",
)
```

Figure 2 : Paramètres pour entrainement

L'entraînement se déroule sur 5 époques, avec une évaluation à chaque époque pour suivre l'évolution des performances du modèle.

5.3.5. Évaluation

Une évaluation est réalisée après l'entraînement en utilisant les métriques suivantes :

Précision (Accuracy)

Score F1 (F1-score)

Perte de distillation

L'évaluation est réalisée avec la fonction compute_metrics qui applique les métriques accuracy et f1 de evaluate.

Une matrice de confusion est également générée pour visualiser la performance du modèle sur chaque classe.

5.3.6. Sauvegarde du modèle

Après l'entraînement, le modèle est sauvegardé pour une utilisation ultérieure. Cette étape permet de charger directement le modèle sans nécessiter un nouvel entraînement.

5.3.7. Déploiement et Utilisation

Le modèle entraîné peut être utilisé pour prédire la classification d'un texte donné. Une interface API ou un script de classification peut être implémenté pour tester le modèle sur de nouvelles données.

6. Résultats et évaluation

6.1. Performance du modèle

L'entraînement du modèle DistilBERT, avec l'assistance d'un modèle professeur (BERT), a permis d'améliorer les performances en tirant parti des connaissances d'un modèle plus robuste tout en réduisant la complexité du modèle final.

6.2. Limites et défis rencontrés

Malgré les bonnes performances du modèle, plusieurs limites et défis ont été identifiés lors du développement et de l'entraînement :

- **Qualité et variété des données** : La précision du modèle est fortement influencée par la qualité du dataset. Un jeu de données plus diversifié pourrait améliorer la robustesse du modèle. Surtout que les datasets des professionnels sont gardé secret, donc non utilisable lors de notre projet.
- **Coût en ressources** : L'entraînement du modèle nécessite une puissance de calcul et d'un espace de stockage important, notamment pour la distillation des connaissances entre BERT et DistilBERT.
- **Généralisation** : Le modèle peut rencontrer des difficultés à généraliser sur des textes non vus, en particulier si le contexte diffère fortement de celui du dataset d'entraînement.

Des améliorations futures pourraient inclure l'augmentation des données et l'exploration de modèles plus légers pour réduire le coût computationnel.

7. Conclusion

Ce projet a permis de mettre en place un modèle performant pour l'identification de contenu lié à la propagande terroriste. Grâce à l'utilisation de modèles de langage avancés comme DistilBERT, nous avons pu obtenir des résultats satisfaisants tout en optimisant les ressources computationnelles nécessaires à l'entraînement.

Malgré les défis rencontrés, les résultats obtenus sont encourageants et démontrent le potentiel dans le domaine de la détection automatique de contenu sensible.

Pour aller plus loin, plusieurs axes d'amélioration peuvent être envisagés, notamment l'enrichissement du jeu de données, l'adoption de méthodes de régularisation pour éviter les biais et l'intégration d'un mécanisme d'explicabilité pour mieux comprendre les décisions du modèle. Ces pistes ouvrent la voie à une application plus robuste et fiable dans des contextes réels.

ANNEXES

TABLE DES TABLEAUX

TABLE DES FIGURES

ANNEXES

Annexe 1 : Arborescence du projet

REFERENCES

[1] https://huggingface.co/

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES