المدرسة الوطنية للعلوم التطبيقية بتطوان
ⵜⵜⵉⵎⵏⵜⵏⵍⵍⵓⵜⵉⵜⵜⵓⵙⵙⵏⵉⵏⵜⵓⵏⵉⵙⵓⵙ
Ecole Nationale des Sciences Appliquées de Tétouan

Filière de Big Data et Intelligence Artificielle
École Nationale des Sciences Appliquées de Tétouan

Rapport de projet

Deepfake Detection

Réalisé par :
Bouhnas Chaymae
EL Alami Nihad
EL Gamani Ahlam

Encadré par :
Prof.Belcaid Anas

Année Universitaire : 2024-2025

Remerciements

« Parfois notre lumière s'éteint, puis elle est
rallumée par un autre être humain.
Chacun de nous doit de sincères
remerciements à ceux qui
ont ravivé leur flamme »
Albert Schweitzer

Après avoir rendu grâce à Dieu le Tout Puissant et le Miséricordieux, nous profitons de l'occasion pour remercier du fond du cœur toute personne qui a contribué de près ou de loin à la réalisation de ce travail.

Nous témoignons notre profonde gratitude à Mr. Belcaid Anas, notre professeur, pour nous avoir accordé l'opportunité d'effectuer ce projet, en tant que notre encadrant, pour ses conseils, son aide et son soutien.

Enfin, nous ne saurions oublier nos familles et proches pour leur soutien moral et leurs encouragements tout au long de cette expérience. Leur présence a été essentielle pour notre réussite.

Abstract

L'émergence de technologies avancées de deepfake représente une menace unique en raison de la capacité du contenu deepfake à échapper à la détection et de son potentiel à créer une réalité alternative. Les deepfakes sont des rendus audio ou vidéo réalistes d'un individu ou d'un groupe, créés à l'aide de l'intelligence artificielle, capables de reproduire les expressions et les intonations d'une personne et de montrer des actions ou des événements qui n'ont jamais eu lieu, comme un candidat à la présidence tenant des propos diffamatoires ou réalisant des activités compromettantes. Les recherches ont montré que les technologies de deepfake ont rapidement progressé en termes d'efficacité et de qualité au cours des trois dernières années. Bien que les technologies de détection des deepfakes deviennent rapidement plus sophistiquées, leur développement semble principalement réactif et repose parfois sur des technologies de création de deepfake obsolètes. Pour résoudre ce problème, nous présentonsun un cadre méthodologique complet basé sur deux architectures : un modèle CNN+LSTM from scratch et un modèle ResNet50+LSTM pré-entraîné. Ces modèles exploitent respectivement des caractéristiques locales et des relations temporelles pour détecter efficacement les deepfakes.

Table des matières

PARTIE 1 : PARTIE THEORIQUE **7**
I. Introduction Générale . . . 7
II. La Détéction de Deep fake . . . 12
1. Les applications de Deepfake Detection . . . 12
2. objectifs de projet . . . 14
3. dataset . . . 14

PARTIE 2 : PARTIE PRATIQUE **16**
I.Préparation des données : . . . 16
Léxtraction : . . . 16
Division des Données en ensemble d'Entraînement et de test . . . 17
Prétraitement et Formatage des Données pour les Modèles . . . 17
1.Architecture de Modèles F rom Scratch : CNN+LSTM . . . 17
1.1 Définition des hyperparamètres et entrée du modèle . . . 18
1.2 Extraction de caractéristiques avec le CNN . . . 18
1.3 Préparation des données pour le réseau récurrent . . . 19
1.4 Réseau LSTM bidirectionnel : une approche avancée . . . 19
1.5 Classification et couches denses . . . 19
Entraînement . . . 19
Compilation et optimisation du modèle . . . 19
Stratégies de régularisation et gestion du surapprentissage . . . 20
Prétraitement des données . . . 20
Entraînement et validation . . . 20
Résultats et analyse des performances . . . 21
Évaluation finale sur l'ensemble de test . . . 21
2. Architecture de modèle ResNet50+LSTM . . . 21
2. Utilisation de ResNet50 comme extracteur de caractéristiques . . . 23
Présentation du modèle ResNet50 . . . 23
Fine-tuning du modèle pré-entraîné . . . 23
3. Intégration du module LSTM pour l'analyse temporelle . . . 23
Transformation des données CNN en séquences temporelles . . . 23
Rôle du LSTM Bidirectionnel . . . 24
4. Post-traitement des données et architecture du modèle final . . . 24
Couches de normalisation et de régularisation . . . 24
Classement final avec softmax . . . 24
5. Optimisation et apprentissage du modèle . . . 24
Optimiseur Adam . . . 24
Fonction de perte et métrique . . . 25
6. Entraînement du modèle . . . 25
Préparation des données d'entrée . . . 25

Encodage one-hot des étiquettes 25
Utilisation des callbacks 25
2.6 Évaluation du modèle 25
L 26
Précision sur l'ensemble de test (*Test Accuracy*) 26
Perte sur l'ensemble de test (*Test Loss*) 27
Analyse qualitative 27
IV. Évolution de la précision et de la perte au cours de l'entraînement 28
1. Modèle CNN+LSTM 28
1. Modèle ResNET+LSTM 29
V. les défis 31
VII. Conclusion 31
VI. Conclusion 35

Table des figures

1 Deepfake principle. 8
2 Illustration of a picture (left) being manufactured (right) utilizing the Deepfake procedure. Note that the manufactured face comes up short on the expressiveness of the first. 9
3 Celeb-DF 14
4 Example frames from the Celeb-DF dataset. The left column is the frame of real videos, and the right five columns are corresponding DeepFake frames generated using a different donor subject. 15
5 Architecture de modèle CNN+LSTM 18
6 Architecture de modèle ResNet+LSTM 22
7 model accuracy and model loss m1 28
8 model accuracy and model loss m2 29
9 l'interface de l'application 31
10 models accuracy overview 32
11 Available Models 32
12 upload your file 32
13 Analysis result for CNN+LSTM 33
14 Extracted Frames 33
15 Upload file againe 33
16 Analysis result for ResNet50+LSTM 34
17 extracted frames 34

Liste des tableaux

1 Summary of Notable Deepfake Tools 12
2 Comparaison des performances des modèles 26
3 table des défis 31

PARTIE 1 : PARTIE THEORIQUE

I. Introduction Générale

Les technologies de modification d'images, de vidéos et d'audios évoluent rapidement. On observe une augmentation des innovations dans les domaines de la transformation des images, des enregistrements sonores et des vidéos. Une large gamme de méthodes pour créer et manipuler du contenu numérique avancé est également disponible. Aujourd'hui, il est possible de générer des images numériques hyperréalistes avec peu de ressources et un simple guide pratique disponible sur internet.

Un deepfake est une technique qui remplace le visage d'une personne spécifique dans une vidéo par celui d'une autre personne. Essentiellement, une image faciale unique est fusionnée avec un portrait intégré. En plus de désigner le résultat final d'une vidéo promotionnelle réaliste, ce terme est également utilisé pour faire référence au produit fini. Outre la création d'images de synthèse extraordinaires (CGI), de réalité virtuelle (VR) et de réalité augmentée (AR), les deepfakes peuvent être utilisés à des fins éducatives, dans l'animation, l'art et le cinéma.

Avec l'évolution des smartphones et l'accès généralisé à une bonne connexion internet, un nombre croissant de réseaux sociaux et de sites de partage de médias a amplifié la capacité de créer et de diffuser des vidéos numériques. Parallèlement, la puissance de calcul à faible coût s'est développée régulièrement, rendant l'apprentissage profond plus performant que jamais. Cependant, ces progrès considérables ont inévitablement engendré de nouveaux défis.

Les deepfakes consistent à manipuler des vidéos et des audios à l'aide de modèles génératifs adversariaux profonds. Il existe de nombreux cas où des deepfakes sont diffusés sur les plateformes de réseaux sociaux, causant des spams et propageant de fausses informations. De tels deepfakes peuvent être terribles et entraîner des situations où des personnes se sentent menacées ou induites en erreur.

Les visages de A sont reproduits par un auto-encodeur EA basé sur un ensemble de données d'images faciales de A, tandis qu'un auto-encodeur EB est utilisé pour reproduire les visages de B à partir d'un ensemble de données d'images faciales de B. Pour créer des images deepfake, il faut assembler les visages ajustés de deux personnes différentes A et B, puis entraîner un auto-encodeur EA pour recréer les visages de A à partir de l'ensemble de données des images faciales de A, ainsi qu'un autre auto-encodeur EB pour B. Les charges d'encodage partagées des auto-encodeurs EA et EB sont utilisées tout en conservant les parties de décodage de chaque encodeur séparées.

En optimisant cet encodeur, toute image contenant le visage de A peut être décodée avec le décodeur de EB, grâce à cet encodeur commun. Ce principe est illustré dans les Figures 1 et 2. Selon cette méthodologie, un encodeur est utilisé pour concaténer

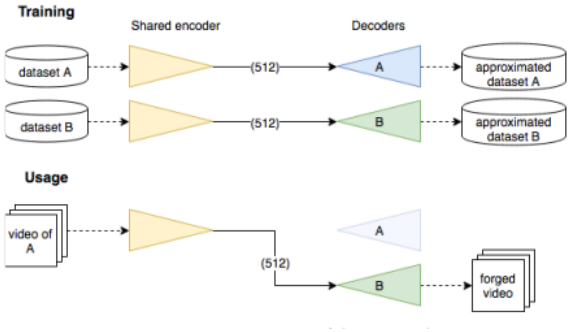

FIGURE 1 – Deepfake principle.

les informations générales liées à l'éclairage, à la position et à l'apparence du visage, tandis qu'un décodeur dédié intègre les détails propres à chaque visage et recrée les traits constants ainsi que les détails. Grâce à cette méthode, les informations pertinentes peuvent être séparées des informations morphologiques. En pratique, les résultats sont excellents, ce qui explique l'importance de cette procédure.

La dernière étape consiste à prendre la vidéo cible, à extraire le visage cible de chaque image, à l'ajuster de manière à ce que l'éclairage et l'expression soient uniformes, puis à utiliser l'auto-encodeur modifié pour produire un autre visage. Ensuite, ce nouveau visage est fusionné avec le visage cible.

FIGURE 2 – Illustration of a picture (left) being manufactured (right) utilizing the Deepfake procedure. Note that the manufactured face comes up short on the expressiveness of the first.

Some other methods for deepfake creation tool

Tools	Links	Key Features
FaceSwap	https://github.com/deepfakes/faceswap	Using two encoder-decoder pairs. Parameters of the encoder are shared.
Faceswap-GAN	https://github.com/shaoanlu/faceswap-GAN	Adversarial loss and perceptual loss (VGGFace) are added to an auto-encoder architecture.
Few-Shot Face Translation	https://github.com/shaoanlu/fewshot-facetranslation-GAN	-Use a pre-trained face recognition model to extract latent embeddings for GAN processing. Incorporates semantic priors obtained by modules from FUNIT and SPADE.
DeepFaceLab	https://github.com/iperov/DeepFaceLab	Expand from the Faceswap method with new models, e.g., H64, H128 , LIAEF128 SAE. Support multiple face extraction models, e.g., S3FD, MTCNN, dlib, or manual.
DFaker	https://github.com/dfaker/df	DSSIM loss function is used to reconstruct face. Implemented based on Keras library.
DeepFake tf	https://github.com/StromWine/DeepFake$_tf$	Similar to DFaker but implemented based on TensorFlow.
AvatarMe	https://github.com/lattas/AvatarMe	Reconstruct 3D faces from arbitrary "in-the-wild" images. Can reconstruct authentic 4K by 6K resolution 3D faces from a single low-resolution image.

MarioNETte	https://hyperconnect.github.io/MarioNETte	A few-shot face reenactment framework that preserves the target identity. No additional fine-tuning phase is needed for identity adaptation.
DiscoFaceGAN	https://github.com/microsoft/DiscoFaceGAN	Generate face images of virtual people with independent latent variables of identity, expression, pose, and illumination. Embed 3D priors into adversarial learning.
StyleRig	https://gvv.mpi-inf.mpg.de/projects/StyleRig	Create portrait images of faces with a rig-like control over a pre-trained and fixed StyleGAN via 3D morphable face models. Self-supervised without manual annotations.
FaceShifter	https://lingxihit.com/FaceShifterPage	Face swapping in high fidelity by exploiting and integrating the target attributes. Can be applied to any new face pairs without requiring subject-specific training.
FSGAN	https://github.com/YuvalNirkin/fsgan	A face swapping and reenactment model that can be applied to pairs of faces without requiring training on those faces. Adjust to both pose and expression variations.

Transformable Bottleneck Networks	https://github.com/kyleoliver/TB-Networks	A method for fine-grained 3D manipulation of image content. Apply spatial transformations in CNN models using a transformable bottleneck.
"Do as I Do" Motion Transfer	https://github.com/carolineec/ Everybody-DanceNow	Automatically transfer the motion from a source to a target person by learning the motion pattern.
Neural Voice Puppetry	https://justusthies.github.io/posts/neural-voice-puppetry	A method for audio-driven facial synthesis. Generate realistic lip syncing and head motion from an audio source using 3D face representations.

TABLE 1: Summary of Notable Deepfake Tools

II. La Détéction de Deep fake

La détection de deepfake est le processus d'utilisation d'algorithmes et de techniques pour identifier si une vidéo ou un autre contenu multimédia a été manipulé ou généré de manière synthétique à l'aide de l'intelligence artificielle (IA), en le classant comme "RÉEL" ou "FAUX". La tâche consiste à concevoir et développer un algorithme d'apprentissage profond pour classer une vidéo comme étant un deepfake ou authentique. Nous prédirons la probabilité que la vidéo soit fausse ou non grâce à la détection de deepfake, qui est généralement une classification binaire où l'entrée est une vidéo (.mp4) et la sortie est une étiquette $L \in \{"REL","FAUX"\}$. L'algorithme analysera la vidéo en entrée pour effectuer cette classification.

1. Les applications de Deepfake Detection

Sécurité et sûreté nationale :

— Menace pour la sécurité mondiale : Les vidéos et images deepfake peuvent être

utilisées comme des armes pour diffuser de la désinformation et inciter des tensions politiques ou religieuses. Par exemple, une vidéo fabriquée montrant un dirigeant politique tenant des propos incendiaires pourrait déstabiliser les relations internationales ou influencer l'opinion publique pendant les élections.

Applications juridiques et médico-légales :

— Identification des preuves falsifiées : Dans les tribunaux, l'authenticité des preuves est cruciale. Les méthodes de détection des deepfakes peuvent garantir que les preuves vidéo ou audio n'ont pas été manipulées, préservant ainsi l'intégrité des processus judiciaires.

— Cyberharcèlement et atteinte à la vie privée : La détection des deepfakes permet d'identifier les vidéos manipulées afin de prévenir les tromperies et la désinformation, protégeant ainsi la réputation et le bien-être des victimes.

Médias sociaux et intégrité publique :

— Prévention de la désinformation et des atteintes à la vie privée : Les plateformes de médias sociaux rencontrent souvent des difficultés pour identifier et supprimer les contenus manipulés. Les systèmes de détection des deepfakes peuvent automatiser le processus de repérage des identités falsifiées ou des médias altérés, garantissant ainsi la confiance des utilisateurs et protégeant leur vie privée.

Industrie et divertissement :

— Amélioration de la vie privée : La technologie deepfake, lorsqu'elle est utilisée de manière éthique, peut masquer les identités et protéger la vie privée dans les médias publics ou sensibles. Par exemple, elle peut anonymiser des individus dans des vidéos de surveillance ou des reportages.

— Essais virtuels de produits : Des applications comme les essais virtuels de maquillage ou les simulations de coiffure s'appuient sur l'échange de visages et la manipulation d'attributs. Les consommateurs peuvent expérimenter des produits virtuellement, améliorant ainsi leur expérience et leur commodité.

— Enjeux éthiques : Malgré ces avantages, l'utilisation abusive de la technologie deepfake pour créer de fausses vidéos de célébrités ou de propagande politique soulève des questions éthiques. Par exemple, la création de faux soutiens par des figures publiques pourrait tromper les audiences.

Santé :

— Intégrité des images médicales : Bien que la détection des deepfakes serve directement à la médecine légale des médias, ses technologies sous-jacentes contribuent à garantir l'authenticité des images médicales. Cela pourrait éviter que des ensembles de données manipulés ou falsifiés ne compromettent les outils de diagnostic alimentés par l'IA.

2. objectifs de projet

Les objectifs du projet incluent le développement d'un modèle capable de détecter et d'étiqueter avec précision les deepfakes. Nous allons utiliser deux modèles :

1. Le premier est un modèle from scratch CNN+LSTM.
2. Le second est un modèle basé sur ResNet50+LSTM.

3. dataset

Bien que les ensembles de données actuels de DeepFake contiennent un nombre suffisant de vidéos, celles-ci présentent divers artefacts visuels qui les distinguent facilement des vidéos réelles. Afin de fournir des données plus pertinentes pour évaluer et soutenir le développement futur des méthodes de détection de DeepFake, l'ensemble de données Celeb-DF a été construit.

FIGURE 3 – Celeb-DF

Bien que les ensembles de données actuels de DeepFake contiennent un nombre suffisant de vidéos, celles-ci présentent divers artefacts visuels qui les distinguent facilement des vidéos réelles. Afin de fournir des données plus pertinentes pour évaluer et soutenir le développement futur des méthodes de détection de DeepFake, l'ensemble de données Celeb-DF a été construit.

L'ensemble de données Celeb-DF se compose de 590 vidéos réelles et de 5 639 vidéos DeepFake (correspondant à plus de deux millions de frames vidéo). La longueur moyenne de toutes les vidéos est d'environ 13 secondes, avec un taux de frame standard de 30 images par seconde. Les vidéos réelles sont choisies parmi des vidéos YouTube publiques, correspondant à des interviews de 59 célébrités, avec une distribution diversifiée en termes de genres, âges et groupes ethniques. 56,8 % des sujets dans les vidéos réelles sont des hommes et 43,2 % sont des femmes. 8,5% ont 60 ans ou plus, 30,5 % ont entre 50 et 60 ans, 26,6 % ont la quarantaine, 28,0 % sont dans la trentaine et 6,4% sont plus jeunes que 30 ans. 5,1 % sont Asiatiques, 6,8 % sont Afro-Américains et 88,1 % sont Caucasiens.

De plus, les vidéos réelles présentent une large gamme de variations dans des aspects tels que la taille des visages des sujets (en pixels), les orientations, les conditions d'éclairage et les arrière-plans. Les vidéos DeepFake sont générées en échangeant les visages pour chaque paire des 59 sujets. Les vidéos finales sont au format MPEG4.0.

FIGURE 4 – Example frames from the Celeb-DF dataset. The left column is the frame of real videos, and the right five columns are corresponding DeepFake frames generated using a different donor subject.

PARTIE 2 : PARTIE PRATIQUE

I.Préparation des données :

L'objectif de cette étape est de préparer un dataset équilibré et structuré à partir de vidéos réelles et synthétiques (fake) en vue de leur utilisation dans un modèle d'apprentissage profond. Les vidéos sources sont organisées dans deux répertoires principaux : l'un contenant les vidéos réelles, et l'autre les vidéos synthétiques. Afin de faciliter la gestion des données et réduire la complexité des calculs, un sous-ensemble de 100 vidéos issues de chaque catégorie est d'abord copié vers des répertoires de travail dédiés. Ces vidéos sont ensuite utilisées pour créer un dataset équilibré contenant 300 vidéos de chaque classe, où les vidéos synthétiques sont sélectionnées aléatoirement pour garantir la diversité ; car Notre dataset présente un déséquilibre entre les classes, avec un plus grand nombre de vidéos deepfake que de vidéos réelles. Ce déséquilibre peut entraîner un biais du modèle en faveur de la classe majoritaire, ce qui affecterait les performances de la détection.

L'éxtraction :

Les vidéos sélectionnées sont ensuite transformées en frames (images individuelles) à l'aide d'une fonction dédiée. Cette fonction extrait un nombre fixe de frames (20 frames par vidéo) tout en assurant une gestion robuste des vidéos corrompues ou incomplètes. Pour cela, les frames sont échantillonnées régulièrement si le nombre de frames est suffisant, ou rééchantillonnées aléatoirement dans le cas contraire. Chaque frame est convertie au format RGB, redimensionnée à une résolution de 64x64 pixels pour réduire la complexité, et normalisée en divisant les valeurs des pixels par 255 afin de stabiliser l'entraînement du modèle. Ces transformations garantissent une représentation uniforme des vidéos et facilitent leur traitement par un modèle d'apprentissage profond.

Division des Données en ensemble d'Entraînement et de test

Pour entraîner et évaluer le modèle, les données sont divisées en deux ensembles : un ensemble d'entraînement (80%) et un ensemble de test (20%). Les vidéos réelles sont étiquetées avec le label 0 et les vidéos synthétiques avec le label 1. Afin d'enrichir le dataset et d'augmenter la taille des données, les frames extraites des vidéos sont transformées en images individuelles. Les frames issues des vidéos réelles et synthétiques sont regroupées séparément, puis combinées pour former un dataset unique, avec des labels associés à chaque image. Les données ainsi générées sont mélangées pour éviter tout biais lié à l'ordre des vidéos et réparties entre l'entraînement et le test.

Prétraitement et Formatage des Données pour les Modèles

Les dimensions finales des ensembles de données sont adaptées aux besoins des modèles d'apprentissage profond : les frames sont organisées sous forme de matrices 4D, avec des dimensions correspondant au nombre d'images, à la hauteur et largeur des frames (64x64 pixels) et au nombre de canaux (3 pour RGB). Cette préparation assure une balance entre les classes, une uniformisation des données et une réduction de la complexité tout en conservant les informations visuelles essentielles. En outre, la flexibilité des données, disponibles sous forme de frames vidéo ou d'images individuelles, permet d'explorer différentes architectures de modèles, comme les CNN ou des combinaisons CNN-LSTM.

II.Les Architectures et Entraînement des Modèles utilisés

1.Architecture de Modèles From Scratch : CNN+LSTM

Dans le cadre de nombreux projets impliquant des données d'images, l'utilisation exclusive de réseaux convolutifs (CNN) permet d'extraire des caractéristiques visuelles pertinentes, mais ne capture pas toujours les relations spatiales ou séquentielles de manière optimale. Une solution prometteuse consiste à combiner un CNN pour l'extraction de caractéristiques locales avec un LSTM, capable d'appréhender les relations à plus grande échelle sous forme de séquences. Cette approche est particulièrement efficace dans des applications où les dépendances spatiales jouent un rôle clé. Dans ce projet, nous pro-

posons notre premier modèle CNN + LSTM destiné à une tâche de classification binaire d'images, ce modèle vise à traiter des images d'entrée de dimensions standardisées et à produire des prédictions de classe précises grâce à une architecture soigneusement conçue et optimisée.

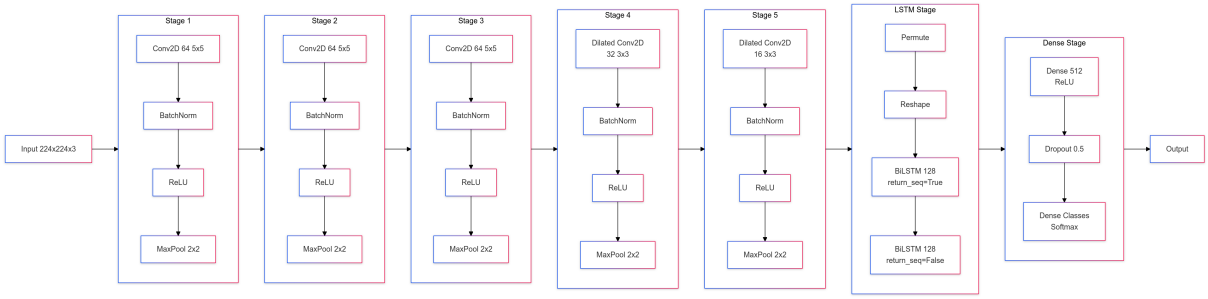

FIGURE 5 – Architecture de modèle CNN+LSTM

1.1 Définition des hyperparamètres et entrée du modèle

La première étape consiste à définir les paramètres essentiels de l'architecture, tels que la forme des images d'entrée et le nombre de classes à prédire : Les images d'entrée ont une taille fixée de (224, 224, 3). Ce choix permet d'harmoniser les dimensions des données d'entrée avec celles de modèles CNN. Dans ce projet, il s'agit d'un problème de classification binaire, donc le nombre de classes est fixé à 2. Une couche de sortie avec une activation softmax est utilisée pour obtenir des probabilités normalisées sur les deux classes possibles.

1.2 Extraction de caractéristiques avec le CNN

L'architecture CNN repose sur une série de couches convolutives suivies de couches de sous-échantillonnage (MaxPooling2D). Chaque couche convolutionnelle applique des filtres de différentes tailles pour détecter les motifs caractéristiques dans l'image. Les premières couches utilisent des filtres de taille (5, 5) avec une activation Relue. Cette fonction d'activation est privilégiée pour sa capacité à introduire de la non-linéarité tout en préservant une propagation efficace du gradient. Après chaque convolution, une normalisation par lot (BatchNormalization) est appliquée afin de stabiliser l'entranîment et d'accélérer la convergence. En complément des couches classiques, deux couches supplémentaires exploitent la convolution dilatée avec un taux de dilatation de 2. Cette technique permet d'élargir le champ réceptif sans augmenter le coût computationnel, offrant ainsi une meilleure captation des informations globales.

1.3 Préparation des données pour le réseau récurrent

Une fois les caractéristiques extraites par le CNN, il est nécessaire de les transformer en séquences exploitables par un LSTM. Cette transformation implique deux opérations principales : Premièrement, une permutation des axes est effectuée afin que la dimension spatiale devienne la séquence d'entrée du LSTM. Deuxièmement, les cartes de caractéristiques sont aplaties pour former une séquence linéaire, où chaque vecteur représente une étape temporelle. Cette préparation permet d'adapter les données d'entrée au format attendu par le réseau récurrent.

1.4 Réseau LSTM bidirectionnel : une approche avancée

L'utilisation d'un LSTM bidirectionnel répond au besoin de capturer les relations séquentielles dans les deux directions : avant pour capter les dépendances contextuelles dans le sens naturel des séquences, et arrière pour renforcer la compréhension des relations réciproques. Deux couches LSTM bidirectionnelles sont empilées : la première retourne une séquence complète permettant à chaque pas temporel de bénéficier d'un contexte enrichi. La seconde retourne un vecteur final condensé, utilisé pour la classification.

1.5 Classification et couches denses

Une fois la séquence traitée par le LSTM, un vecteur final est obtenu. Ce vecteur passe ensuite par une couche dense de 512 neurones avec activation ReLU. Cette couche dense permet d'apprendre des représentations complexes de haut niveau. Enfin, une couche de sortie à 2 neurones avec activation softmax est utilisée pour produire les probabilités des deux classes. L'activation softmax garantit que la somme des probabilités des deux classes soit égale à 1.

Entraînement

Compilation et optimisation du modèle

La compilation du modèle est réalisée avec l'optimiseur Adam, connu pour sa robustesse et sa capacité à ajuster dynamiquement le taux d'apprentissage. Le taux d'apprentissage initial est fixé à 0.0001. La fonction de perte choisie est l'entropie croissée catégorielle (categorical crossentropy), adaptée à la classification multi-classe avec des étiquettes en-

codées en one-hot. La métrique accuracy est utilisée pour suivre l'évolution de la précision du modèle au cours de l'entranîment.

Stratégies de régularisation et gestion du surapprentissage

Pour améliorer la généralisation du modèle et éviter le surapprentissage, deux stratégies complémentaires sont employées : Premièrement, une réduction dynamique du taux d'apprentissage est mise en place via le callback ReduceLROnPlateau. Si la perte de validation ne s'améliore pas pendant 3 époques consécutives, le taux d'apprentissage est réduit de moitié, permettant ainsi au modèle de converger plus finement vers un minimum local. Deuxièmement, l'arrêt anticipé (EarlyStopping) est utilisé pour stopper automatiquement l'entranîment si la perte de validation ne s'améliore plus pendant 5 époques consécutives. Cette stratégie prévient l'entranîment excessif et permet d'économiser des ressources computationnelles.

Prétraitement des données

Avant l'entranîment du modèle, les images doivent être correctement prétraitées : Premièrement, les images sont redimensionnées à (224, 224) afin de correspondre à la taille d'entrée du modèle. Deuxièmement, les valeurs des pixels sont normalisées entre 0 et 1 en divisant par 255. Cette normalisation stabilise l'entranîment et améliore la convergence. Enfin, les étiquettes de classe sont converties en vecteurs one-hot, une représentation adaptée à la fonction de perte utilisée.

Entraînement et validation

Le modèle est entraîné sur un ensemble de données d'entraînement et évalué sur un ensemble de données de validation. L'entraînement est effectué sur 20 époques avec une taille de batch de 32. Cette configuration permet un bon compromis entre la rapidité de l'entraînement et la stabilité de la convergence. Le callback ReduceLROnPlateau ajuste dynamiquement le taux d'apprentissage en fonction de la performance du modèle sur les données de validation, évitant ainsi un taux d'apprentissage trop élevé lorsqu'un plateau de performance est atteint. En parallèle, le callback EarlyStopping surveille la perte sur l'ensemble de validation et arrête automatiquement l'entraînement si cette dernière ne s'améliore pas au bout de 5 époques consécutives. Cette approche garantit que le modèle

conserve les meilleurs poids obtenus durant l'entraînement, ce qui minimise le risque de surapprentissage tout en optimisant l'utilisation des ressources de calcul.

Résultats et analyse des performances

Une fois l'entraînement terminé, il est crucial d'évaluer les performances du modèle sur un ensemble de test indépendant. Cela permet de mesurer la capacité du modèle à généraliser sur de nouvelles données. L'analyse des courbes d'entraînement et de validation, incluant l'évolution de la perte et de la précision au fil des époques, est essentielle pour vérifier la convergence correcte du modèle et détecter d'éventuels problèmes de surapprentissage .

Évaluation finale sur l'ensemble de test

Une fois l'entraînement terminé et les meilleurs poids du modèle retenus, il est essentiel de procéder à une évaluation finale sur un ensemble de test indépendant. Cette étape permet de valider la capacité du modèle à généraliser sur des données qu'il n'a jamais vues. Avant l'évaluation, les images de l'ensemble de test sont prétraitées de manière identique aux images d'entraînement. Elles sont redimensionnées à la taille d'entrée requise par le modèle, soit (224, 224), puis les valeurs des pixels sont normalisées entre 0 et 1 en les divisant par 255. Cette normalisation est cruciale pour garantir la cohérence des données d'entrée et améliorer la stabilité des prédictions du modèle. Une fois les données prêtes, l'évaluation du modèle est réalisée à l'aide de la méthode evaluate, qui retourne la perte et la précision sur l'ensemble de test. Ces deux métriques fournissent des indications quantitatives sur la performance finale du modèle. Les résultats obtenus (perte et précision) permettent de juger de la qualité du modèle sur de nouvelles données. Une faible perte et une précision élevée indiquent que le modèle a bien appris les caractéristiques pertinentes sans trop s'adapter aux données d'entraînement, ce qui est un bon indicateur de généralisation.

2. Architecture de modèle ResNet50+LSTM

Le modèle hybride ResNet50-LSTM combine deux architectures puissantes pour la classification d'images : le ResNet50, un modèle de réseau de neurones convolutionnel (CNN) pré-entraîné sur ImageNet, et un Long Short-Term Memory (LSTM), un type de réseau récurrent spécialisé dans l'analyse de données séquentielles. L'idée derrière ce

modèle est d'exploiter les caractéristiques extraites par ResNet50 tout en ajoutant une capacité d'apprentissage temporel via les LSTM, ce qui est pertinent dans des scénarios où les relations temporelles ou spatiales doivent être capturées.

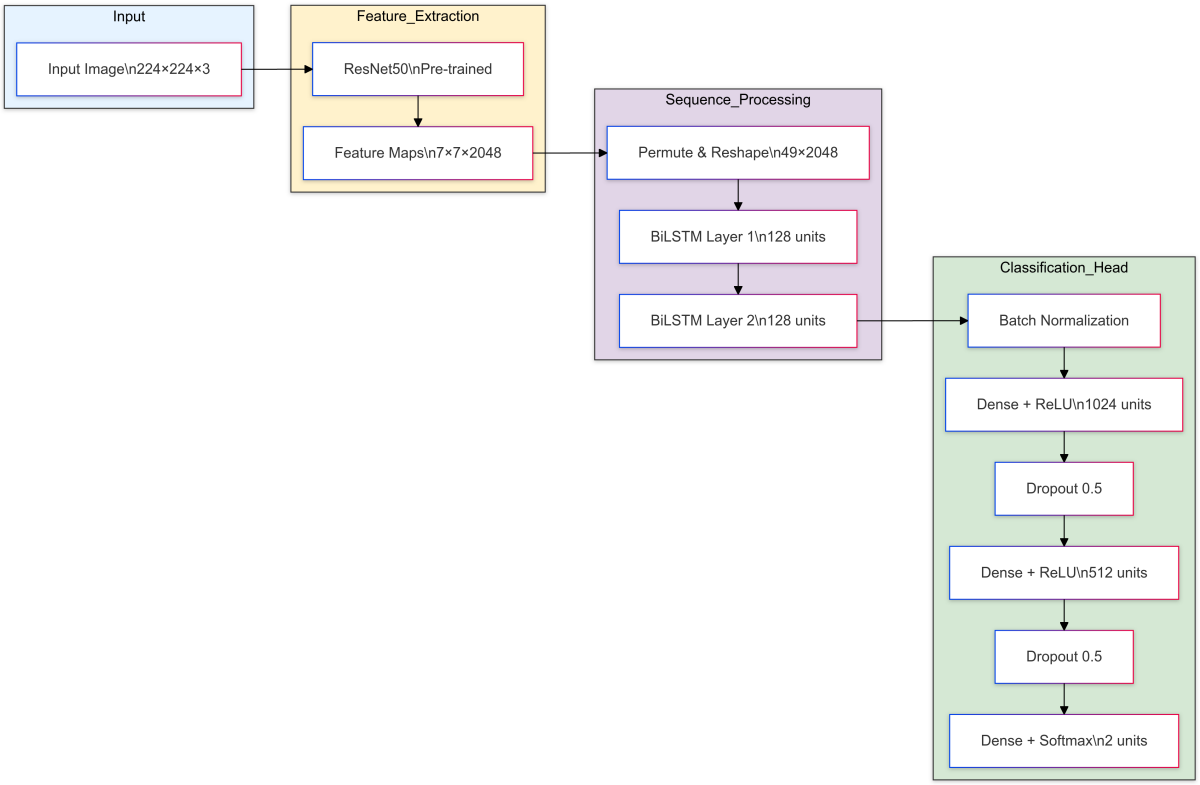

FIGURE 6 – Architecture de modèle ResNet+LSTM

2.1 Utilisation de ResNet50 comme extracteur de caractéristiques

Présentation du modèle ResNet50

Le modèle ResNet50 est une version allégée du ResNet, qui est un réseau de neurones profond utilisant des blocs de résidus (Residual Blocks) pour atténuer le problème de l'explosion du gradient dans les réseaux profonds. Grâce à cette architecture, ResNet50 est capable de former des modèles très profonds tout en maintenant une bonne stabilité pendant l'entraînement. Nous exploitons ici ResNet50 pré-entraîné sur le dataset ImageNet, ce qui permet de capitaliser sur des représentations de haut niveau déjà apprises, réduisant ainsi le besoin d'une grande quantité de données étiquetées pour la phase d'entraînement spécifique à la tâche.

Fine-tuning du modèle pré-entraîné

Un fine-tuning sélectif a été appliqué sur les couches supérieures du modèle ResNet50 pour ajuster le modèle pré-entraîné aux spécificités de notre propre jeu de données. Seules les 50 dernières couches ont été débloquées, permettant au modèle de réapprendre certains des derniers filtres tout en maintenant la structure d'apprentissage générale déjà acquise sur ImageNet. Cela permet de réduire le nombre de paramètres à entraîner, tout en optimisant la performance pour notre tâche spécifique.

2.2 Intégration du module LSTM pour l'analyse temporelle

Transformation des données CNN en séquences temporelles

Une fois que ResNet50 a extrait les caractéristiques spatiales de l'image d'entrée, ces dernières sont transformées pour être compatibles avec les réseaux récurrents comme LSTM. Cette étape est réalisée en permutant et redimensionnant la sortie de ResNet50 de manière à la convertir en une séquence d'entrées temporelles. Concrètement, les dimensions spatiales de l'image sont comprimées en un vecteur de caractéristiques de taille (49, 2048), représentant un espace de 7x7 pixels avec 2048 caractéristiques par pixel. Ce vecteur est ensuite introduit dans un LSTM bidirectionnel qui permet de capturer à la fois les dépendances passées et futures dans les données d'entrée.

Rôle du LSTM Bidirectionnel

L'utilisation d'un LSTM bidirectionnel permet d'exploiter les relations temporelles dans les deux directions (avant et arrière). Ce choix est crucial dans des tâches comme la reconnaissance d'images, où la séquence d'informations visuelles est interprétée dans un contexte global, prenant en compte à la fois les informations présentes et futures pour chaque région de l'image.

2.3 Post-traitement des données et architecture du modèle final

Couches de normalisation et de régularisation

Une normalisation par lots (BatchNormalization) est utilisée après les couches LSTM pour stabiliser l'entraînement et accélérer la convergence du modèle. Ensuite, plusieurs couches Dense avec activations ReLU sont ajoutées pour introduire de la non-linéarité et permettre au modèle de capturer des relations complexes. Dropout est appliqué à chaque couche dense pour éviter l'overfitting, particulièrement dans des modèles complexes où il y a un risque élevé de surapprentissage sur les données d'entraînement.

Classement final avec softmax

La couche finale est une couche dense avec activation softmax, permettant de classifier les images en fonction de la probabilité d'appartenance à chaque classe. Le nombre de neurones dans cette couche est égal au nombre de classes de la tâche de classification (2 dans ce cas), et la sortie représente la probabilité que chaque entrée appartienne à chacune des classes.

2.4. Optimisation et apprentissage du modèle

Optimiseur Adam

L'optimisation du modèle est réalisée avec l'optimiseur Adam, un algorithme d'optimisation qui combine les avantages des algorithmes AdaGrad et RMSProp. Il est particulièrement adapté aux problèmes de réseaux profonds et de grandes dimensions, et a montré une bonne efficacité dans la convergence des modèles CNN et LSTM.

Fonction de perte et métrique

La fonction de perte utilisée est la categorical crossentropy, car il s'agit d'un problème de classification multi-classes. Le modèle est évalué à l'aide de la métrique accuracy, qui mesure le pourcentage de prédictions correctes sur l'ensemble des données.

2.5 Entraînement du modèle

Préparation des données d'entrée

Les images sont redimensionnées à une taille uniforme de (224, 224) pour correspondre à l'entrée de ResNet50 et normalisées à une plage de [0, 1] en divisant les valeurs de pixels par 255. Cela est crucial pour garantir la stabilité et l'efficacité de l'entraînement.

Encodage one-hot des étiquettes

Les étiquettes sont encodées en one-hot encoding, une méthode courante dans les problèmes de classification multi-classes, où chaque étiquette est représentée par un vecteur binaire avec une valeur "1" à l'indice correspondant à la classe de l'exemple et "0" pour les autres classes.

Utilisation des callbacks

Afin d'optimiser le processus d'entraînement, plusieurs callbacks sont utilisés : ReduceLROnPlateau, qui réduit dynamiquement le taux d'apprentissage en cas de stagnation de la perte de validation, et EarlyStopping, qui arrête l'entraînement si la performance ne s'améliore pas après un certain nombre d'époques, tout en rétablissant les meilleurs poids observés.

2.6 Évaluation du modèle

Une fois l'entraînement terminé, le modèle est évalué sur l'ensemble de test. Les résultats sont mesurés en termes de perte et d'exactitude, offrant une indication claire de la performance du modèle sur des données qu'il n'a pas vues auparavant.

Performances obtenues

e modèle est évalué sur l'ensemble de test pour obtenir la perte et l'accuracy. Ces deux métriques permettent d'estimer à la fois la précision du modèle (en termes de classification correcte) et la qualité de la prédiction.

III.La Comparaison Entre Modèles

Le tableau ci-dessous présente les résultats obtenus pour ces deux modèles sur un jeu de données de test, en utilisant 20 frames par séquence. Les métriques évaluées sont la précision du modèle sur le jeu de test (*Test Accuracy*) ainsi que la perte associée (*Test Loss*). Ces métriques permettent de comparer la capacité des modèles à généraliser sur des données inconnues.

Modèles	**Nombre de Frames**	**Test Accuracy**	**Test Loss**
Model **From Scratch** (CNN + LSTM)	20	78%	0.3
ResNet50 + LSTM	20	93%	0.18

TABLE 2 – Comparaison des performances des modèles

1. Précision sur l'ensemble de test (*Test Accuracy*)

— **CNN+LSTM (créé de zéro)** : 78%

Une précision de 78% reflète une performance modérée. Le modèle, bien que capable d'apprendre certaines caractéristiques discriminantes, semble moins robuste aux variations complexes des deepfakes. Cette performance peut être attribuée à une capacité d'extraction de caractéristiques limitée par rapport à un modèle pré-entraîné comme ResNet50.

— **ResNet50+LSTM** : 93%

Une précision de 93% démontre une nette supériorité en termes de capacité à détecter les deepfakes. Cette amélioration s'explique principalement par l'efficacité de ResNet50 à capturer des caractéristiques de bas et haut niveau grâce à son entraînement préalable sur une grande base de données.

2. Perte sur l'ensemble de test (*Test Loss*)

— **CNN+LSTM (créé de zéro)** : 0.30

Une perte de 0.30 suggère que le modèle a une marge d'erreur plus importante, ce qui est cohérent avec sa précision plus faible. Ce résultat peut également indiquer une difficulté à converger vers un optimum global, probablement à cause de la complexité inhérente à l'apprentissage des représentations à partir de zéro.

— **ResNet50+LSTM** : 0.18

La perte plus faible de 0.18 confirme une meilleure capacité du modèle à généraliser sur des données non vues. ResNet50 fournit des représentations initiales de haute qualité, ce qui aide le LSTM à apprendre les relations temporelles de manière plus efficace et avec une meilleure convergence.

3. Analyse qualitative

— **CNN+LSTM (créé de zéro)**

Bien que performant, le modèle CNN+LSTM souffre de certaines limitations, notamment la nécessité d'un ajustement manuel précis des hyperparamètres et d'un temps d'entraînement plus long en raison de la conception non optimisée de l'architecture CNN. De plus, en l'absence d'un pré-entraînement, le modèle dépend entièrement des données d'entraînement fournies, ce qui peut entraîner un sous-apprentissage des détails subtils des manipulations vidéo caractéristiques des deepfakes.

— **ResNet50+LSTM**

Ce modèle bénéficie d'un transfert d'apprentissage grâce à ResNet50, ce qui lui permet d'identifier efficacement des caractéristiques complexes, même dans des situations où les artefacts des deepfakes sont peu visibles. La robustesse du modèle se traduit par une précision et une perte bien meilleures, ainsi qu'une capacité accrue à détecter différentes catégories de manipulations vidéo.

IV. Évolution de la précision et de la perte au cours de l'entraînement

1. Modèle CNN+LSTM

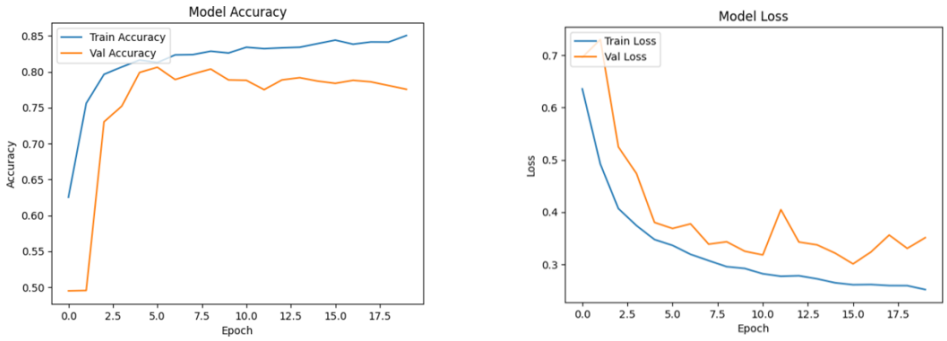

FIGURE 7 – model accuracy and model loss m1

Les graphiques montrent l'évolution des métriques d'entraînement et de validation au fil des époques pour le modèle CNN + LSTM. Le premier graphique illustre la progression de la précision (Accuracy) sur les ensembles d'entraînement et de validation, tandis que le second montre la diminution de la perte (Loss) pour les deux ensembles.

Analyse des graphiques

1. **Graphique de la précision :**

   On observe une augmentation rapide de la précision sur l'ensemble d'entraînement au cours des premières époques, suivie d'une stabilisation autour de 85 %. La précision sur l'ensemble de validation suit une tendance similaire, atteignant environ 80 %, ce qui indique une bonne généralisation du modèle.

2. **Graphique de la perte :**

   La perte d'entraînement diminue de manière continue, traduisant un apprentissage efficace. Quant à la perte de validation, elle diminue également mais présente des fluctuations à partir de la 10^{e} époque, ce qui pourrait suggérer un début de surapprentissage (overfitting).

   Toutefois, la différence entre les pertes d'entraînement et de validation reste modérée.

Ces résultats démontrent que le modèle parvient à apprendre efficacement les caractéristiques discriminantes tout en maintenant une bonne capacité de généralisation sur les données de validation.

1. Modèle ResNet50+LSTM

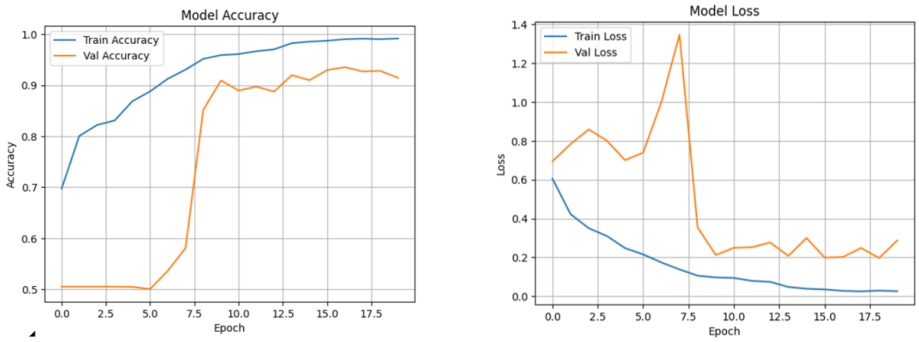

FIGURE 8 – model accuracy and model loss m2

Les graphiques ci-dessus présentent l'évolution des métriques d'entraînement et de validation au fil des époques pour le modèle (ResNet50+LSTM). Le premier graphique montre l'évolution de la précision (Accuracy) sur les ensembles d'entraînement et de validation, tandis que le second illustre la diminution de la perte (Loss) pour ces deux ensembles.

Analyse des graphiques

1. **Graphique de la précision :**

   La précision sur l'ensemble d'entraînement augmente rapidement au cours des premières époques, atteignant un plateau autour de 98 %. La précision de l'ensemble de validation montre une augmentation plus progressive, se stabilisant autour de 90 % après environ 10 époques.

   L'écart entre les courbes d'entraînement et de validation indique que le modèle ResNet50 apprend bien les caractéristiques des données d'entraînement, mais il existe un risque de surapprentissage (overfitting), car la précision de validation plafonne tandis que celle de l'entraînement continue d'augmenter.

2. **Graphique de la perte :**

   La perte d'entraînement diminue régulièrement, indiquant une amélioration continue

du modèle. Cependant, la perte de validation présente une augmentation significative autour de la 7^{e} époque, suivie d'une diminution et de fluctuations. La perte de validation ne diminue pas de manière aussi régulière que la perte d'entraînement. Les fluctuations de la perte de validation pourraient indiquer que le modèle commence à surapprendre les données d'entraînement à partir d'une certaine époque.

V. les défis

Défi	Description	Impact	Solution
Manque de ressources GPU	L'entraînement de modèles CNN et LSTM sur des vidéos nécessite une grande capacité de calcul, notamment des GPU puissants.	Limite la vitesse d'entraînement et rend difficile l'expérimentation avec différents paramètres et architectures.	Utilisation de Kaggle pour l'entraînement, qui offre des ressources GPU pour les projets.
Résolution des images	Les vidéos deepfake peuvent être manipulées à différentes résolutions, ce qui impacte la qualité des caractéristiques extraites.	Des résolutions faibles rendent la détection plus difficile, tandis que des résolutions plus élevées nécessitent plus de ressources.	Aucune solution trouvée pour ce défi spécifique.
Adaptation des fonctions d'extraction	Les vidéos doivent être correctement traitées pour extraire les bonnes caractéristiques avant d'être introduites dans les modèles.	Les fonctions d'extraction doivent être adaptées à chaque type de vidéo et de modèle pour garantir la précision.	Tests de plusieurs méthodes d'extraction pour trouver celle qui fonctionne le mieux avec le modèle.

TABLE 3 – table des défis

VII. application

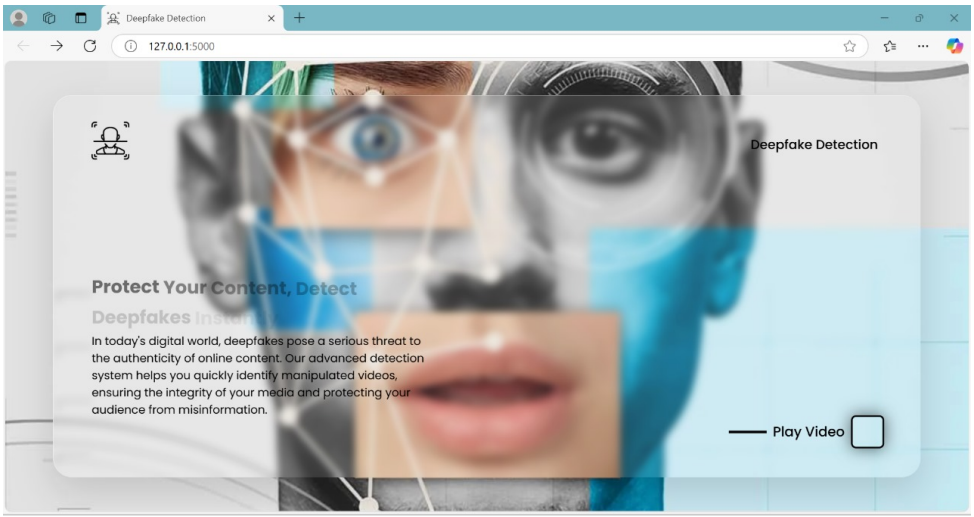

FIGURE 9 – l'interface de l'application

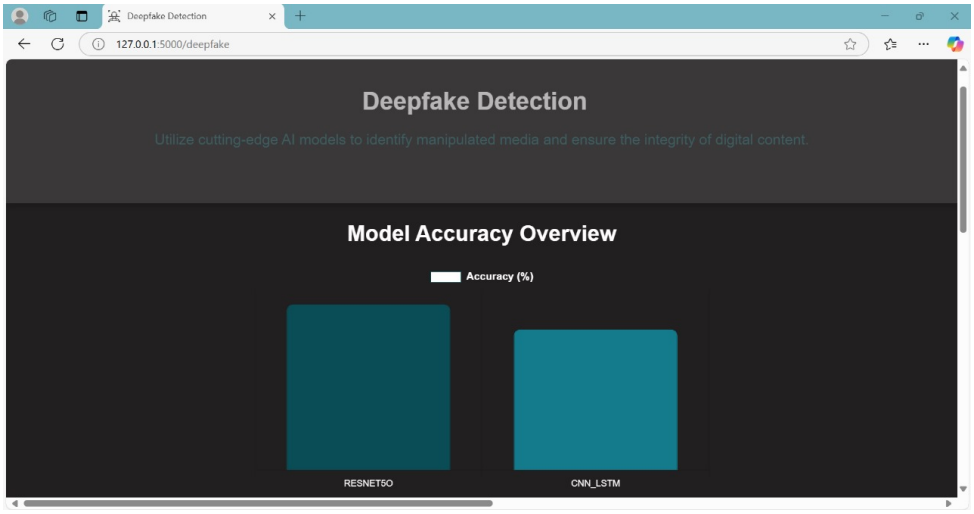

FIGURE 10 – models accuracy overview

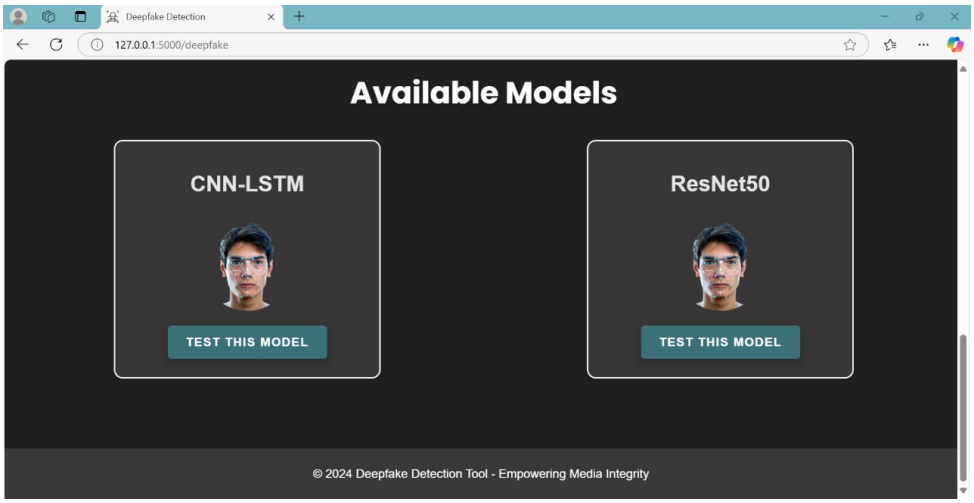

FIGURE 11 – Available Models

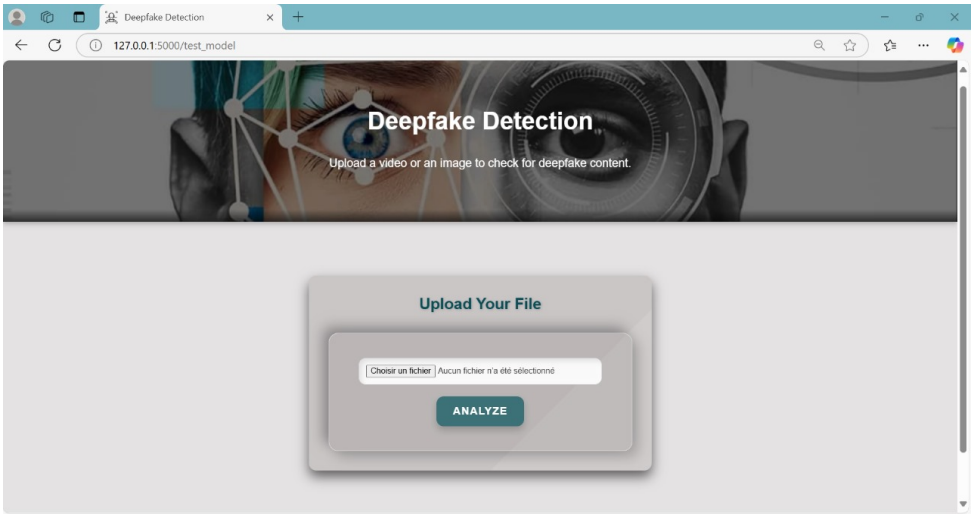

FIGURE 12 – upload your file

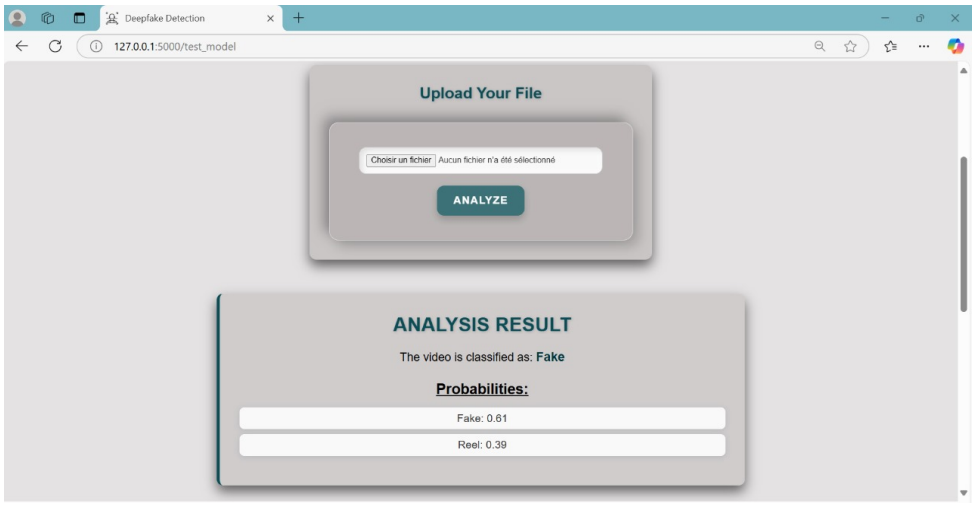

FIGURE 13 – Analysis result for CNN+LSTM

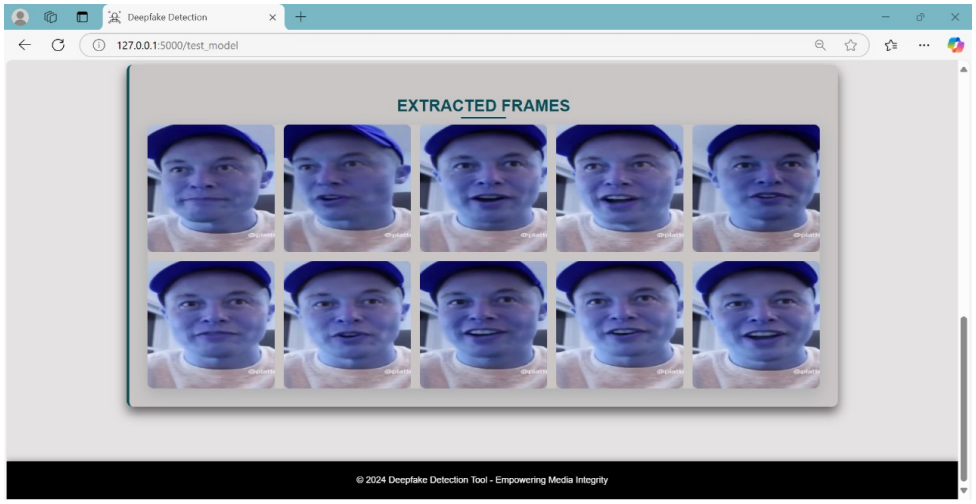

FIGURE 14 – Extracted Frames

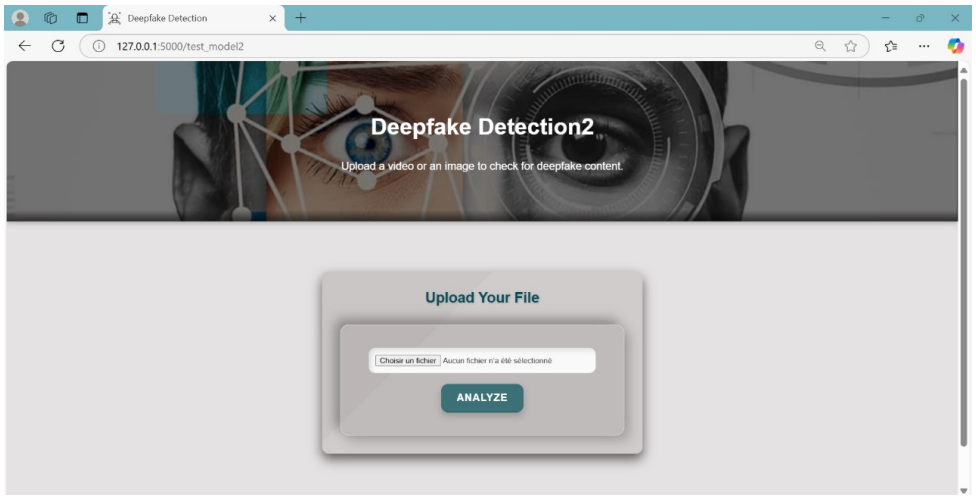

FIGURE 15 – Upload file againe

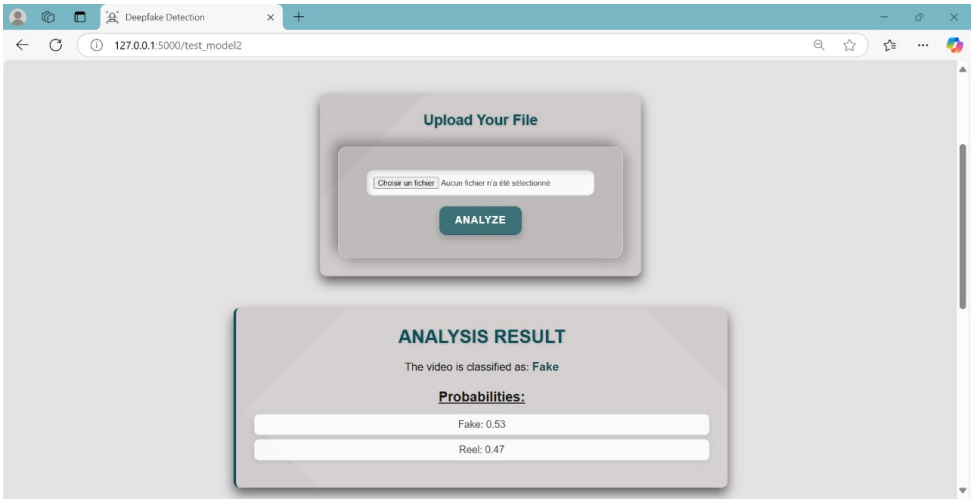

FIGURE 16 – Analysis result for ResNet50+LSTM

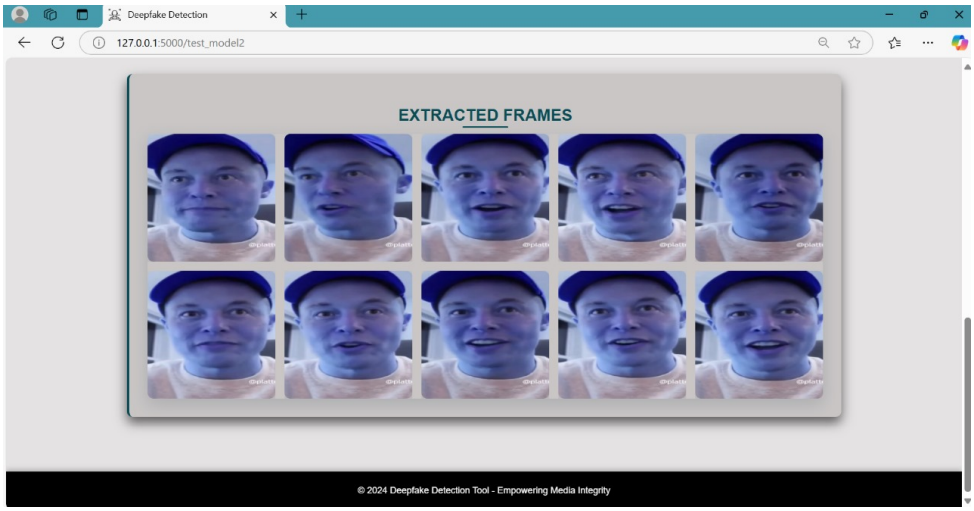

FIGURE 17 – extracted frames

VI. Conclusion

La détection des deepfakes est devenue un enjeu crucial dans un contexte où la manipulation de contenus numériques prolifère, posant des défis majeurs pour la sécurité, la vie privée et l'intégrité de l'information. Ce projet a exploré deux approches complémentaires pour relever ces défis : un modèle CNN+LSTM développé de zéro et un modèle ResNet50+LSTM pré-entraîné. Les résultats obtenus démontrent la supériorité du modèle ResNet50+LSTM, avec une précision de 93%, grâce à sa capacité à tirer parti de caractéristiques pré-apprises et à analyser les relations temporelles de manière efficace.

Cependant, des défis subsistent, notamment en termes de gestion des ressources de calcul et d'adaptation des modèles à des résolutions variées. Malgré ces limitations, les avancées réalisées dans ce projet mettent en lumière le potentiel des approches hybrides et ouvrent la voie à des applications concrètes, notamment dans les domaines de la cybersécurité et de la lutte contre la désinformation. L'intégration future d'ensembles de données plus diversifiés et de techniques d'optimisation avancées pourrait encore améliorer les performances et la robustesse de ces modèles.