Université Paris Cité

Détection d'images truquées par Stable Diffusion

Par :
Samy Aghiles AOUABED
Malek Mohamed BENNABI

Date : Mai 2024

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Les avancées technologiques en intelligence artificielle ont révolutionné la création et la manipulation de contenus visuels, donnant naissance à des outils sophistiqués comme les deepfakes. Ces technologies permettent de générer des images et des vidéos d'une qualité si réaliste qu'il devient difficile de distinguer le vrai du faux. Si les deepfakes peuvent offrir des applications bénéfiques, ils présentent également des risques considérables, notamment en matière de désinformation, de manipulation de l'opinion publique et de violation de la vie privée. La détection de ces contenus manipulés est donc devenue une priorité pour assurer la confiance numérique et la véracité de l'information.

Ce rapport se concentre sur la détection des deepfakes générés par des modèles avancés tels que Stable Diffusion. Nous avons exploré différentes approches pour résoudre ce problème complexe : une approche directe basée sur des réseaux de neurones convolutionnels (CNN), une approche fréquentielle utilisant le spectre de Fourier, et une approche de réduction de dimensionnalité basée sur des autoencodeurs. Chaque méthode a été testée et évaluée pour déterminer son efficacité et sa capacité à généraliser face à divers générateurs de deepfakes.

Le rapport est structuré comme suit : nous commençons par présenter la problématique et le contexte de notre étude, suivis par une revue de l'état de l'art des techniques de génération et de détection de deepfakes. Ensuite, nous détaillons la conception de nos approches, avant de décrire leur implémentation. Enfin, nous présentons et analysons les résultats obtenus, en concluant sur les points forts et les limites de nos méthodes ainsi que les perspectives d'amélioration.

Chapitre 1

Problématique et contexte

1.1 Introduction

Les avancées technologiques en intelligence artificielle ont permis la création de contenus visuels d'une qualité sans précédent. Parmi ces avancées, les deepfakes se distinguent par leur capacité à générer des images et des vidéos d'apparence authentique. Ces technologies, en permettant la manipulation sophistiquée de contenus visuels, posent des défis significatifs en matière de véracité de l'information et de confiance numérique. Dans un monde où les contenus visuels jouent un rôle crucial dans la formation de l'opinion publique et la prise de décisions, la capacité de détecter ces manipulations devient essentielle.

Les deepfakes, notamment, ont un potentiel malveillant qui suscite de grandes préoccupations. Ils peuvent être utilisés pour diffuser de la désinformation, manipuler l'opinion publique, harceler des individus, et même influencer des décisions politiques. Face à cette menace, l'identification des médias manipulés représente un défi technique majeur, exacerbé par l'évolution rapide de ces technologies. Ce défi nécessite une vigilance constante et un investissement substantiel dans le développement d'outils de détection robustes et efficaces.

1.2 Problématique

Le projet présenté dans ce rapport vise à concevoir une approche pour la détection des images deepfake, en se concentrant particulièrement sur celles générées par des modèles comme Stable Diffusion. Nous utilisons les bibliothèques TensorFlow, Keras ainsi que pytorch pour le développement et la mise en œuvre de notre solution, en explorant et en évaluant diverses méthodes de détection pour différencier entre les images réelles authentiques et les images truquées.

1.3 Dataset

Pour réaliser ce projet, nous avons utilisé plusieurs jeux de données qu'on va exposer par la suite :

1.3.1 FakeFace Dataset[1]

DeepFakeFace (DFF) est un ensemble de données d'images truquées qui contient 30 000 paires d'images réelles et de fausses images. Les 30 000 images réelles de l'ensemble de données proviennent toutes de l'ensemble de données IMDB-WIKI. L'ensemble de données se compose de 120 000 images, dont 30 000 images réelles et 90 000 images truquées. Les images fakes sont générées par trois modèles génératifs différents pour synthétiser les deepfakes : Stable Diffusion v1.5, Stable Diffusion Inpainting et une puissante boîte à outils InsightFace[2].

— **IMDB-WIKI :** Ce jeu de données contient des images de visage des 100 000 acteurs les plus populaires figurant sur le site web IMDb labelisées avec la date de naissance, le nom, le sexe et toutes les images liées à cette personne. Grace aux pages Wikipedia de ces acteurs, l'âge biologique (réel) de chacune de ces images a été attribué.
Au total, ce dataset contient 460 723 images de visages de 20 284 célébrités à partir d'IMDb et 62 328 à partir de Wikipedia, soit 523 051 au total.

FIGURE 1.1 – Deux images reelles du dataset IMDB-WIKI

— **Inpainting :** Contient 30 000 images fakes générées par le modèle Inpainting Stable Diffusion .

FIGURE 1.2 – Trois images fakes du dataset Inpainting

— **Text2img :** Contient 30 000 images fakes générées par Stable Diffusion v1.5 .

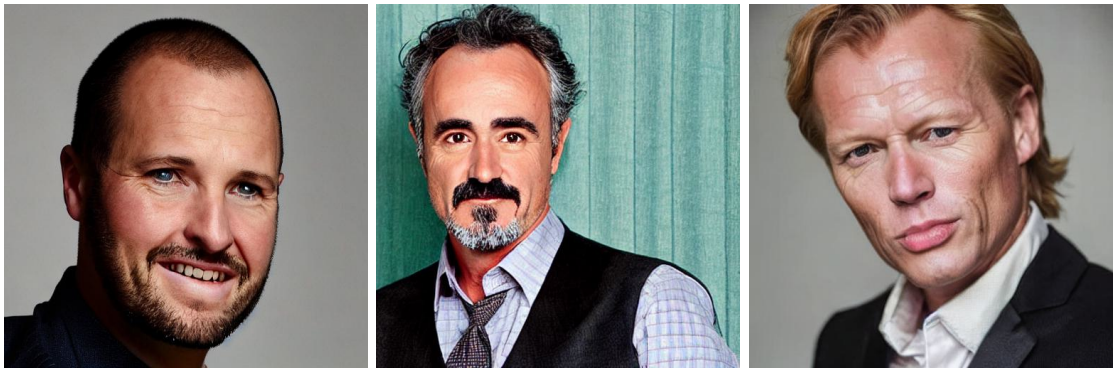

FIGURE 1.3 – Trois photos fakes du dataset text2img

— **Insight :** Contient 30 000 images fakes générées par la toolbox InsightFace.

FIGURE 1.4 – Trois photos fakes du dataset insight

1.3.2 CelebA Dataset

CelebA (CelebFaces Attributes Dataset) est un ensemble de données à grande échelle d'attributs de visages comprenant plus de 200 000 images de célébrités de taille (178×218), chacune comportant 40 annotations d'attributs. Les images de cet ensemble de données couvrent de grandes variations de pose et des arrière-plans bruités. CelebA présente une grande diversité, de grandes quantités et de riches annotations, notamment :

— 10 177 personnes différentes
— 202 599 images de visages
— 5 emplacements de repères, 40 annotations d'attributs binaires par image.

L'ensemble de données peut être utilisé comme ensembles d'entraînement et de test pour les tâches de vision par ordinateur.[3]

FIGURE 1.5 – Deux photos réelles du dataset CelebA

1.3.3 140k Real and Fake faces

Ce jeu de données est composé des 70k vrais visages du jeu de données Flickr collecté par Nvidia, ainsi que 70k faux visages échantillonnés à partir du dataset '1 Million Fake Faces' (Ces images ont été générées par StyleGAN).[4]

Dans ce jeu de données, les deux datasets précédemment cités ont été combinés, leurs images redimensionnées en 256px, et divisés en données de train, validation et test. Le dataset contient également quelques fichiers CSV pour plus de commodité.[5]

FIGURE 1.6 – Une photo réelle (droite) et une photo fake du dataset 140k Real and Fake faces

1.4 Conclusion

Dans ce chapitre, nous avons présenté une vue générale sur notre thème et la problématique du sujet abordé. Nous avons commencé par introduire notre projet et la problématique, ensuite nous avons exploré les différents jeux de données utilisés des fausses et vraies images.
Dans le chapitre qui va suivre, nous allons aborder les techniques de génération d'images truquées ainsi que quelques méthodes pour détecter et différencier les images réelles des images fausses.

Chapitre 2

État de l'Art

2.1 Introduction

Dans le chapitre précédent, nous avons introduit la problématique de notre projet, ainsi que les jeux de données utilisé dans la tâche de création et détection d'images truquées. Dans ce chapitre, nous passerons en revue les différentes technologies de génération d'images artificielles, connues sous le nom de modèles de deepfake. Ces technologies ont révolutionné la manière dont les images et les vidéos peuvent être générées et manipulées, posant à la fois des opportunités et des défis importants, notamment en terme de détection de contenu manipulé. Nous examinerons les principaux modèles utilisés dans ce domaine, en mettant l'accent sur leurs architectures, leurs mécanismes d'entraînement et leurs applications.

2.2 Modèles de Deepfake

2.2.1 GANs (Generative Adversarial Networks)[6]

Les réseaux génératifs adversaires (GANs) ont été introduits par Ian Goodfellow et al. en 2014 dans l'article intitulé *"Generative Adversarial Nets"*. Les GANs représentent une avancée révolutionnaire dans la génération de données synthétiques grâce à leur architecture innovante composée de deux réseaux de neurones : un générateur et un discriminateur.

Le générateur G prend un vecteur de bruit z échantillonné d'une distribution $p_z(z)$ et produit des échantillons de données synthétiques $G(z)$. Le discriminateur D reçoit des échantillons réels x de la distribution de données $p_{data}(x)$ et des échantillons synthétiques $G(z)$, et il émet une probabilité $D(x)$ indiquant si l'échantillon est réel ou généré. Les deux réseaux sont entraînés simultanément via une procédure d'optimisation où le générateur tente de minimiser la probabilité que le discriminateur le détecte, tandis que le discriminateur tente de maximiser cette probabilité. Cette confrontation est formulée comme suit :

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Les contributions majeures des GANs incluent :

— **Cadre Adversaire** : L'utilisation d'un cadre adversaire pour l'entraînement des modèles génératifs, permettant une amélioration continue des performances de génération.
— **Formulation Mathématique** : Une formulation mathématique détaillée du processus d'entraînement, incluant les fonctions de coût pour les deux réseaux.
— **Résultats Expérimentaux** : Des résultats expérimentaux montrant que les GANs peuvent générer des images réalistes à partir de bruit aléatoire, surpassant les méthodes précédentes en termes de qualité visuelle.

2.2.2 Autoencoders Variationnels (VAE)[7]

Les Autoencoders Variationnels (VAE), proposés par Kingma et Welling en 2013 dans l'article *"Auto-Encoding Variational Bayes"*, combinent les principes de l'apprentissage profond et des probabilités bayésiennes pour la génération de données. Un VAE encode les données d'entrée dans une distribution latente probabiliste, permettant de générer de nouvelles données en échantillonnant cette distribution. Cette approche permet de capturer la variabilité des données dans l'espace latent et de générer des échantillons similaires mais distincts.

Le VAE utilise un encodeur $q_\phi(z|x)$ pour mapper les données d'entrée x à une distribution latente z, et un décodeur $p_\theta(x|z)$ pour reconstruire les données à partir de cette distribution latente. La fonction de perte du VAE combine une perte de reconstruction et une régularisation par divergence Kullback-Leibler (KL) :

$$\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] + \mathrm{KL}(q_\phi(z|x)||p(z))$$

Les points clés des VAE incluent :

— **Encodage et Décodage Probabilistes** : Utilisation de réseaux de neurones pour encoder les données dans une distribution latente et décoder les échantillons de cette distribution en données de sortie.

— **Régularisation par Divergence KL** : Introduction d'une régularisation par divergence Kullback-Leibler (KL) pour aligner la distribution latente avec une distribution normale, facilitant l'échantillonnage.

— **Applications et Expériences** : Des expériences démontrant l'efficacité des VAE pour la génération d'images et la reconstruction de données, avec des exemples sur des ensembles de données de référence comme MNIST.

2.2.3 StyleGAN[8]

StyleGAN, proposé par Karras et al. en 2019 dans l'article *"A Style-Based Generator Architecture for Generative Adversarial Networks"*, introduit une architecture innovante pour les GANs, permettant un contrôle fin sur les caractéristiques des images générées via une manipulation basée sur le style. L'architecture inclut un réseau de mappage qui transforme les vecteurs de bruit en vecteurs de style, influençant différents niveaux de la génération d'images.

La nouveauté de StyleGAN réside dans son architecture basée sur le style, où un vecteur de style w est échantillonné à partir d'un espace latent $\mathcal{W}$ et injecté à chaque couche de génération à l'aide d'un module AdaIN (Adaptive Instance Normalization). Ce mécanisme permet de moduler les caractéristiques des images à différents niveaux, offrant ainsi un contrôle hiérarchique :

$$y = \mathrm{AdaIN}(x, y) = \gamma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta(y)$$

Les aspects importants de StyleGAN incluent :

— **Contrôle Hiérarchique** : Division de la génération d'images en plusieurs couches, chacune influencée par des vecteurs de style différents, permettant une manipulation précise des caractéristiques de l'image.

— **Synthèse de Haute Qualité** : Capacité à générer des images de très haute qualité, réalistes et détaillées, surpassant les méthodes précédentes en termes de fidélité visuelle.

— **Applications** : Utilisation dans la génération de visages humains synthétiques, avec des résultats impressionnants sur des ensembles de données tels que FFHQ (Flickr-Faces-HQ).

2.2.4 Stable Diffusion[9]

Stable Diffusion, introduit par Rombach et al. en 2022 dans l'article *"High-resolution image synthesis with latent diffusion models"*, propose une méthode de génération d'images haute réso-

lution basée sur les modèles de diffusion latente. Stable Diffusion utilise un processus de diffusion inversée, où des images bruitées sont progressivement améliorées pour devenir des visuels nets et détaillés.

Le modèle de diffusion latente fonctionne en appliquant un processus de bruitage progressif aux données d'entrée, puis en entraînant un modèle pour inverser ce processus. Le processus de diffusion est modélisé comme une chaîne de Markov, où à chaque étape, une image légèrement plus bruitée est générée :

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Le modèle est ensuite entraîné pour prédire l'image sans bruit à partir de l'image bruitée, permettant de générer des images haute résolution en inversant le processus de diffusion.

Les points saillants de Stable Diffusion incluent :

— **Processus de Diffusion** : Utilisation d'un processus de diffusion stochastique pour transformer progressivement des images bruitées en images claires, en entraînant le modèle à inverser cette diffusion.

— **Haute Résolution** : Capacité à générer des images de haute résolution avec des détails fins et réalistes, surpassant les techniques de génération traditionnelles en termes de qualité visuelle.

— **Applications Diverses** : Potentiel pour diverses applications, allant de la création artistique à la conception graphique, en fournissant un outil puissant pour la génération de médias numériques.

2.3 Architectures CNN pour la Détection de Deepfake

2.3.1 VGG16[10]

L'article intitulé *"Very Deep Convolutional Networks for Large-Scale Image Recognition"* par Simonyan et Zisserman (2014) introduit l'architecture VGG16, qui se distingue par sa profondeur et sa simplicité. VGG16 comprend 16 couches, incluant des couches de convolution et de fully connected, utilisant uniquement des convolutions 3x3 avec des pas de 1 et des max-pooling 2x2 avec des pas de 2. Cette conception standardise la taille des cartes de caractéristiques, facilitant ainsi la mise en œuvre et l'optimisation.

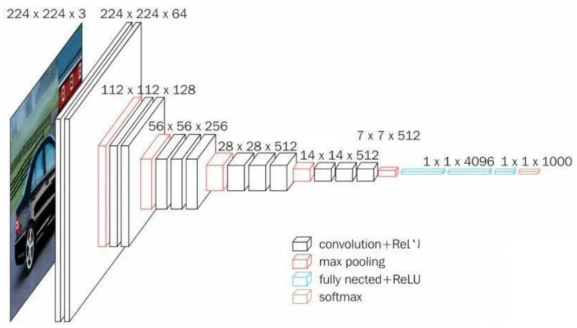

FIGURE 2.1 – Architecture de VGG16[11]

Les principales contributions et caractéristiques de VGG16 sont :

1. **Profondeur accrue** : VGG16 se compose de 13 couches de convolution suivies de 3 couches entièrement connectées. La profondeur accrue permet au modèle d'apprendre des représentations de plus en plus abstraites et discriminantes des données d'entrée, améliorant les performances de classification, notamment sur des ensembles de données complexes comme ImageNet.

2. **Utilisation de petites convolutions 3x3** : L'utilisation systématique de filtres de taille 3x3 permet de capturer efficacement les caractéristiques spatiales tout en maintenant un nombre raisonnable de paramètres. Cette approche permet également un empilement de couches plus profondes sans augmenter de manière significative la charge computationnelle.

3. **Max-Pooling pour la réduction dimensionnelle** : Les couches de max-pooling 2x2 permettent de réduire les dimensions des cartes de caractéristiques, facilitant ainsi l'apprentissage et réduisant le risque de surapprentissage. Cette technique conserve les informations les plus importantes tout en éliminant les redondances.

4. **Performances élevées** : VGG16 a démontré des performances robustes sur le jeu de données ImageNet, atteignant une précision top-5 de 92.7%. Cette architecture a également servi de base pour de nombreuses recherches ultérieures en vision par ordinateur, en raison de sa conception modulaire et de sa capacité à généraliser sur divers ensembles de données.

2.3.2 DenseNet[12]

L'article *"Densely Connected Convolutional Networks"* par Huang et al. (2016) présente DenseNet, une architecture innovante qui introduit une connectivité dense entre les couches. Contrairement aux réseaux traditionnels où chaque couche a des connexions uniquement avec la couche suivante, DenseNet connecte chaque couche à toutes les couches précédentes. Cela signifie que chaque couche reçoit en entrée les cartes de caractéristiques de toutes les couches antérieures.

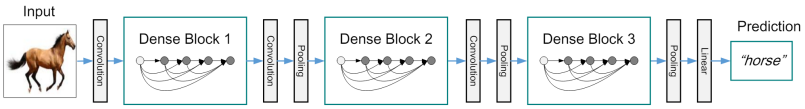

FIGURE 2.2 – Un DenseNet profond avec trois blocs denses. Les couches entre deux blocs adjacents sont appelées couches de transition et modifient les tailles des cartes de caractéristiques via la convolution et le pooling.[12]

Les principales contributions et caractéristiques de DenseNet sont :

1. **Connectivité dense** : DenseNet établit des connexions directes entre chaque couche et toutes les couches précédentes, facilitant ainsi la propagation des gradients et l'apprentissage des caractéristiques. Cette architecture réduit le problème de gradient évanescent, couramment rencontré dans les réseaux profonds.

2. **Réduction des paramètres** : En partageant les caractéristiques entre les couches, DenseNet réduit considérablement le nombre de paramètres. Chaque couche produit une carte de caractéristiques moins redondante, ce qui permet d'avoir des réseaux plus compacts et efficaces sans compromettre les performances.

3. **Efficacité de l'apprentissage** : Les réseaux densément connectés favorisent la réutilisation des caractéristiques apprises, ce qui améliore l'efficacité de l'apprentissage et la généralisation. Cette approche agit également comme une forme de régularisation, réduisant le risque de surapprentissage.

4. **Bloc de transition** : DenseNet utilise des blocs de transition composés de couches de convolution et de pooling pour contrôler la complexité du modèle et ajuster les dimensions des cartes de caractéristiques. Ces blocs permettent de gérer efficacement les tailles des caractéristiques tout en maintenant des performances élevées.

5. **Performances expérimentales** : DenseNet a été évalué sur plusieurs jeux de données de référence tels que CIFAR-10, CIFAR-100, SVHN et ImageNet. Les résultats montrent une amélioration significative par rapport aux architectures précédentes. Par exemple, sur CIFAR-10, DenseNet-BC (L=100, k=12) atteint une erreur de 4.51%, surpassant les performances de nombreux modèles contemporains tout en utilisant moins de paramètres.

2.4 Conclusion

Dans ce chapitre, nous avons exploré diverses technologies et architectures de réseaux de neurones convolutionnels (CNN) utilisées pour la détection de deepfake. Nous avons commencé par une introduction aux modèles de deepfake, en mettant en lumière les contributions majeures des architectures telles que les GANs, les Autoencoders Variationnels, StyleGAN et Stable Diffusion. Ensuite, nous avons approfondi deux architectures CNN spécifiques : VGG16 et DenseNet.

Ces architectures sont particulièrement pertinentes pour la détection de deepfake en raison de leur capacité à extraire des caractéristiques complexes et discriminantes des images. Leur efficacité et leurs performances élevées en font des choix solides pour développer des modèles robustes de détection de deepfake.

Dans ce travail, nous allons nous concentrer sur la détection de deepfakes générés par Stable Diffusion, en utilisant les architectures CNN étudiées pour développer des méthodes robustes et efficaces de détection.

Le chapitre suivant se concentrera sur la conception de notre approche.

Chapitre 3

Conception des approches Proposées

3.1 Introduction

Le chapitre précédent a examiné l'état de l'art de la création d'images truquées ainsi que certaines méthodes utilisées pour leur détection. Dans ce chapitre, nous allons détailler notre propre approche pour la détection de deepfakes générés par Stable Diffusion. Nous diviserons cette discussion en trois sections principales : l'approche directe, l'approche basée sur le spectre de Fourier, et l'approche de réduction de dimensionnalité.

3.2 Approche Directe

Dans cette section, nous explorerons l'utilisation d'un modèle de deep learning CNN classique pour détecter les deepfakes directement à partir des données brutes. Nous entraînerons le modèle sur les ensembles de données inpainting et Wiki, car le sous-ensemble inpainting est le plus pertinent pour notre objectif de détection de deepfakes.

3.2.1 Description de l'approche

L'approche directe implique l'entraînement d'un réseau de neurones convolutionnels (CNN) sur les images des ensembles de données inpainting et wiki. L'objectif est de construire un classificateur capable de distinguer les images réelles des images truquées.

3.2.2 Architecture du modèle

Les modèles CNN utilisés sont basés sur les architectures VGG16 et DenseNet, modifiées pour inclure un global pooling et une couche de sortie avec une activation sigmoid.

3.2.3 Entraînement et validation

Le modèle sera entraîné sur un ensemble d'entraînement, validé sur un ensemble de validation, et évalué sur un ensemble de test final. De plus, les performances du modèle seront évaluées sur les ensembles de données InsightFace, Text2Image et 140k Real and Fake Faces générés par GAN.

3.3 Approche Basée sur le Spectre de Fourier

Cette section examine l'utilisation du spectre de Fourier des images comme entrée pour un modèle CNN. Cette approche est motivée par le fonctionnement de Stable Diffusion, qui consiste à débruiter les images en plusieurs étapes successives.

3.3.1 Description de l'approche

L'approche basée sur le spectre de Fourier consiste à transformer les images en leur représentation fréquentielle avant de les passer à travers un modèle CNN. Cette transformation permet de capturer des artefacts spécifiques souvent présents dans les deepfakes mais invisibles dans le domaine spatial.

3.3.2 Transformation en spectre de Fourier

Chaque image est convertie en son spectre de Fourier à l'aide de la transformation de Fourier rapide (FFT). Le spectre résultant est utilisé comme entrée pour le modèle CNN.

3.3.3 Entraînement et validation

Le modèle sera entraîné, validé et testé sur les mêmes ensembles de données que l'approche directe, avec des spectres de Fourier des images. Les performances seront comparées à celles obtenues avec l'approche directe.

3.4 Approche de Réduction de Dimensionnalité

Cette section explore l'utilisation de techniques de réduction de dimensionnalité pour extraire des embeddings des images, qui sont ensuite utilisés pour entraîner un modèle de classification.

3.4.1 Description de l'approche

L'approche de réduction de dimensionnalité vise à réduire la complexité des images tout en préservant les informations discriminantes. Nous utilisons des autoencodeurs que nous avons entraînés à reconstruire les visages de célébrités du dataset CelebA.

3.4.2 Techniques de réduction de dimensionnalité

Les autoencodeurs sont des réseaux de neurones conçus pour apprendre une représentation comprimée des données d'entrée. Ils sont entraînés sur le dataset CelebA pour obtenir des embeddings compacts des images.

3.4.3 Entraînement et validation

Les embeddings obtenus par les autoencodeurs seront utilisés pour entraîner un modèle de classification binaire. Le modèle sera entraîné, validé et testé sur les mêmes ensembles de données que les approches précédentes. Les performances seront comparées à celles des autres approches.

3.5 Conclusion

Dans ce chapitre, nous avons détaillé la conception de notre approche pour la détection de deepfakes générés par Stable Diffusion. Nous avons exploré trois approches différentes : l'approche directe, l'approche basée sur le spectre de Fourier, et l'approche de réduction de dimensionnalité. Chaque méthode sera testée et comparée pour évaluer leur efficacité relative. Le prochain chapitre se concentrera sur la mise en œuvre de ces approches.

Chapitre 4

Implémentation

4.1 Introduction

Dans le chapitre précédent, nous avons détaillé la conception de notre approche pour la détection de deepfakes générés par Stable Diffusion. Nous avons exploré trois approches différentes : l'approche directe, l'approche basée sur le spectre de Fourier, et l'approche de réduction de dimensionnalité. Dans ce chapitre, nous allons aborder l'implémentation de ces approches. Nous commencerons par décrire les environnements matériel et logiciel utilisés, puis nous présenterons en détail l'implémentation de chaque approche.

4.2 Environnements

4.2.1 Environnement matériel

Nous avons utilisé principalement 4 machines lors du développement :

1. Google Colab Free Tier
   — Système d'exploitation : Ubuntu 20.04.5 LTS (notebooks).
   — CPU : Intel(R) Xeon(R) CPU @ 2.00GHz.
   — RAM : 12GB.
   — GPU : Nvidia Tesla T4 VRAM :15GB.
2. Kaggle
   — Système d'exploitation : Ubuntu 20.04.6 LTS (notebooks).
   — CPU : Intel(R) Xeon(R) CPU @ 2.00GHz.
   — RAM : 29GB.
   — GPU : Nvidia P100 VRAM :16GB.

4.2.2 Environnement logiciel

— Python[13] : est un langage de programmation interprété et multi paradigme de haut niveau. Il est doté d'un typage dynamique fort. Il possède une gestion automatique de la mémoire. Sa simplicité d'utilisation le rend populaire dans le milieu scientifique et plus particulièrement dans le domaine du machine learning.
   — **TensorFlow :** est une librairie de développement de modèles de machine et deep learning open source développé par Google. Il permet de développer des solutions de machine learning en utilisant des API de bas niveau[14].
   — **Keras :** est une API de haut niveau de la librairie TensorFlow[15].
   — **OpenCV :** est une librairie open source pour le machine learning et la vision par ordinateur. Elle est implémenté en python, C++, Java et matlab[16].

— **sklearn[17] :** est une librairie qui implémente un large panel d'algorithmes de machine learning pour la résolution de problèmes supervisés et non supervisés en fournissant des fonctions de haut niveau.

4.3 Distribution des données

L'image ci-dessous illustre la distribution spatiale des datasets après réduction de dimensionnalité via l'analyse en composantes principales (PCA). Comme le montre le graphique, les datasets réels et synthétiques ne forment pas des clusters distincts mais sont plutôt entremêlés les uns avec les autres. Cette observation suggère que tout modèle capable de distinguer ces ensembles ne repose pas sur la différenciation des datasets en tant que tels, mais apprend plutôt les caractéristiques intrinsèques des images permettant de distinguer les deepfakes des images réelles. En d'autres termes, le modèle développe une compréhension fine des traits caractéristiques des images, ce qui lui permet de discriminer efficacement entre les images authentiques et les deepfakes.

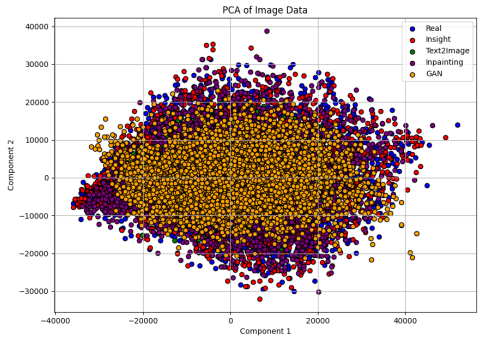

FIGURE 4.1 – Distribution 2D

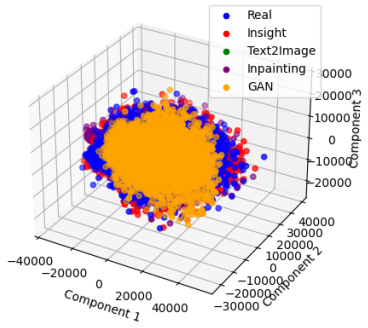

FIGURE 4.2 – Distribution 3D

FIGURE 4.3 – Distribution de la donnée Deepfake et réelle dans l'espace après réduction PCA

4.4 Approche directe

Pour la détection des deepfakes, nous avons opté pour le fine-tuning et l'entraînement de modèles CNN basés sur des architectures pré-entrainées. Le dataset principalement utilisé pour l'entraînement est le sous-ensemble Inpainting, car il correspond le mieux à l'objectif de notre projet de détection de deepfakes.

Nous avons entraîné au total cinq modèles CNN. Le tableau ci-dessous résume les différences entre ces modèles :

— **VGG16_V1**
  — **Learning Rate** : 0.001
  — **Algorithme d'Optimisation** : Adam
  — **Nombre de Couches Gelées** : Toutes les couches du modèle VGG16 de base
  — **Batch Size** : 128
  — **Nombre d'Epochs** : 10
  — **Classification Head** : MLP de 256 neurones
— **VGG16_V2**
  — **Learning Rate** : 0.001
  — **Algorithme d'Optimisation** : Adam

    - **Nombre de Couches Gelées** : Toutes les couches du modèle VGG16 de base
    - **Batch Size** : 128
    - **Nombre d'Epochs** : 10
    - **Classification Head** : MLP de 2048 neurones
- **VGG16_V3**
    - **Learning Rate** : 0.0001
    - **Algorithme d'Optimisation** : Adam
    - **Nombre de Couches Gelées** : Aucune
    - **Batch Size** : 64
    - **Nombre d'Epochs** : 5
    - **Classification Head** : Global average pooling 2D
- **DenseNet_V1**
    - **Learning Rate** : 0.0001
    - **Algorithme d'Optimisation** : Adam
    - **Nombre de Couches Gelées** : Aucune
    - **Batch Size** : 64
    - **Nombre d'Epochs** : 25
    - **Classification Head** : Global average pooling 2D
- **DenseNet_V2**
    - **Learning Rate** : 0.0001
    - **Algorithme d'Optimisation** : Adam
    - **Nombre de Couches Gelées** : Aucune
    - **Batch Size** : 64
    - **Nombre d'Epochs** : 3
    - **Classification Head** : Global average pooling 2D

4.5 Approche Spectre de fourier

4.6 Approche Basée sur le Spectre de Fourier

Pour cette approche, nous avons sélectionné DenseNet_V2 en raison de sa capacité à converger rapidement. Le dataset a été généré en suivant les étapes suivantes :

- **Conversion en Niveaux de Gris** : Chaque image a été convertie en niveaux de gris pour simplifier le traitement.
- **Application de la Transformée de Fourier en 2D** : La Transformée de Fourier en 2D a été appliquée sur chaque image en niveaux de gris.
- **Centrage de la Fréquence Zéro** : La composante de fréquence zéro a été centrée dans le spectre de Fourier pour une meilleure visualisation.
- **Calcul du Spectre de Magnitude** : Le spectre de magnitude a été calculé en prenant le logarithme de la valeur absolue de la transformée de Fourier, afin de rendre les variations plus perceptibles.
- **Enregistrement des Images de Magnitude** : Les images résultantes du spectre de magnitude ont été enregistrées et utilisées comme données d'entrée pour le modèle DenseNet_V2.

Cette transformation permet de capturer des caractéristiques spécifiques souvent présentes dans les deepfakes mais invisibles dans le domaine spatial, fournissant ainsi une base solide pour la détection des manipulations d'images.

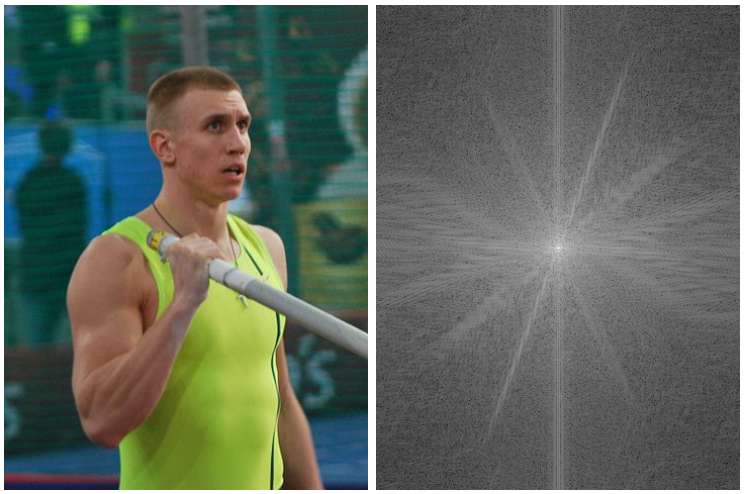

FIGURE 4.4 – Une image réelle (à gauche) et son spectre de Fourier

4.7 Approche Réduction de dimensions

Pour cette approche, nous avons utilisé un autoencodeur [18] entraîné sur 60 000 images du dataset CelebA. L'objectif était de développer un modèle capable de produire des embeddings compacts et informatifs des images de visages, utilisables pour la classification binaire afin de distinguer les images réelles des deepfakes. Voici les détails de l'architecture et de l'implémentation :

4.7.1 Entraînement de l'Autoencodeur

L'architecture de notre autoencodeur [18] est conçue pour réduire progressivement les dimensions des images d'entrée et capturer les caractéristiques essentielles dans un espace latent de 2048 dimensions. L'autoencodeur se compose d'un encodeur et d'un décodeur. L'encodeur transforme les images en un vecteur latent, tandis que le décodeur reconstruit les images à partir de ce vecteur latent :

Architecture de l'Encodeur :
- **Dimensions Latentes :** 2048
- **Taille d'Entrée :** (64, 64, 3)
- **Profondeur :** 1
- Conv2D (256 filtres, 3x3, stride 2, same padding)
- LeakyReLU (alpha=0.2)
- BatchNormalization
- Conv2D (128 filtres, 3x3, stride 2, same padding)
- LeakyReLU (alpha=0.2)
- BatchNormalization
- Conv2D (64 filtres, 3x3, stride 2, same padding)
- LeakyReLU (alpha=0.2)
- BatchNormalization
- Flatten
- Dense (2048)

Architecture du Décodeur :
- Dense (produit des dimensions du volume de sortie de l'encodeur)
- Reshape

— Conv2DTranspose (64 filtres, 3x3, stride 2, same padding)
— LeakyReLU (alpha=0.2)
— BatchNormalization
— Conv2DTranspose (128 filtres, 3x3, stride 2, same padding)
— LeakyReLU (alpha=0.2)
— BatchNormalization
— Conv2DTranspose (256 filtres, 3x3, stride 2, same padding)
— LeakyReLU (alpha=0.2)
— BatchNormalization
— Conv2DTranspose (3 filtres, 3x3, same padding)
— Activation (sigmoid)

L'autoencodeur complet a été compilé avec une fonction de perte de l'erreur quadratique moyenne (MSE) et optimisé avec Adam (learning rate de 1e-2).

4.7.2 Utilisation de l'Encodeur pour la Classification Binaire

Après avoir entraîné l'autoencodeur, nous avons extrait la partie encodeur pour l'utiliser afin de générer des embeddings. Ces embeddings ont ensuite été utilisés comme entrée pour un modèle de classification binaire. Ce modèle a été conçu en ajoutant une couche dense avec une activation sigmoïde à la sortie de l'encodeur.

Architecture du Modèle de Classification Binaire :
— **Entrée :** Embeddings de l'encodeur
— **Couche de Sortie :** Dense (1 neurone, activation sigmoïde)

Le modèle a été compilé avec une fonction de perte de l'entropie croisée binaire (*binary cross-entropy*) et optimisé avec Adam (learning rate de 1e-4).

4.7.3 Préparation des Sous-Datasets de Visages

Pour améliorer la qualité des données d'entraînement et de test, nous avons utilisé MTCNN[19] (Multi-task Cascaded Convolutional Networks) pour détecter et extraire les visages des images. Deux sous-datasets ont été créés : un pour les visages réels et un pour les visages deepfakes. Cela nous permet de nous concentrer uniquement sur les régions pertinentes des images et d'améliorer la précision du modèle.

Cette approche permet de capturer les caractéristiques essentielles des images de visages dans un espace latent compact, facilitant ainsi la classification des images en réelles ou en deepfakes.

4.8 Conclusion

Dans ce chapitre, nous avons décrit l'implémentation des trois approches pour la détection de deepfakes générés par Stable Diffusion. Nous avons présenté les environnements matériel et logiciel utilisés, et détaillé l'approche directe avec des modèles CNN pré-entraînés, l'approche basée sur le spectre de Fourier, et l'approche de réduction de dimensionnalité avec autoencodeurs.

Le prochain chapitre se concentrera sur les résultats et l'évaluation de ces modèles, en analysant leurs performances à l'aide de diverses métriques telles que la précision, le rappel, et le F1-score.

Chapitre 5

Résultats et évaluations

5.1 Introduction

Dans le chapitre précédent, nous avons détaillé les trois approches que nous avons utilisé ;l'approche directe avec modele CNN, l'approche basée sur le spectre de Fourier, et l'approche de réduction de dimensionnalité.

Dans ce chapitre, nous allons présenter les différents résultats d'évaluation de ces modèles, puis on procédera à l'analyse de leurs performances à l'aide d'une variété de métriques à savoir la précision, le rappel, et le F1-score.

Pour chacune des approches suivantes nous allons tester sur 10 000 images (5000 fakes et 5000 réelles), ensuite nous commenterons les résultats obtenus.

5.2 Approche directe

Modèle	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Auc**
VGG16_V1	50.10	/	/	/	/
VGG16_V2	51.30	/	/	/	/
VGG16_V3	95.85	96.14	95.54	94.73	99.11
DenseNet_V2	97.75	96.64	98.94	97.87	99.65

TABLE 5.1: Comparaison des différents modèles selon divers critères sur le dataset Inpainitng

Comme le montre le tableau 5.1, l'ajout de la couche GlobalAveragePooling a permis d'obtenir de meilleurs résultats avec VGG16_V3 et DenseNet_V2, ce dernier modèle atteignant de meilleures performances tout en convergeant plus rapidement.

Les tableaux 5.2, 5.3 et 5.4 ci-dessous illustrent les performances de nos modèles, entraînés sur le dataset Inpainting, lorsqu'ils sont évalués sur des jeux de données qu'ils n'ont jamais vus et qui ont été générés par d'autres techniques de génération de deepfakes.

Modèle	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Auc**
VGG16_V3	55.67	53.15	95.64	54.38	68.11
DenseNet_V2	55.71	53.06	98.88	69.06	69.84

TABLE 5.2: Comparaison des différents modèles selon divers critères sur le dataset Insight

Le tableau 5.2 montre que nos modèles ont des performances proches du hasard, indiquant qu'ils ne se sont pas suffisamment généralisés pour détecter ce type de deepfake.

Modèle	Accuracy	Precision	Recall	F1-Score	Auc
VGG16_V3	93.24	91.79	94.98	93.47	98.07
DenseNet_V2	95.10	92.35	99.02	95.56	99.22

Table 5.3: Comparaison des différents modèles selon divers critères sur le dataset Text2Img

Le tableau 5.3 montre que les modèles présentent des performances similaires pour le dataset Text2Image généré à partir de modèles Stable Diffusion.

Modèle	Accuracy	Precision	Recall	F1-Score	Auc
VGG16_V3	57.18	54.06	95.54	55.57	64.35
DenseNet_V2	53.95	52.08	98.98	52.99	68.47

Table 5.4: Comparaison des différents modèles selon divers critères sur le dataset 140k

Enfin, le tableau 5.4 montre que le modèle performe très mal sur les images générées par GAN.

En résumé, les performances des modèles VGG16_V3 et DenseNet_V2 sur différents datasets révèlent plusieurs points clés. Sur le dataset Inpainting, les deux modèles montrent une bonne capacité à capturer les caractéristiques des images générées. Cependant, sur le dataset Insight, les performances sont nettement inférieures pour les deux modèles, ce qui s'explique par le fait que Insight utilise le face swapping, une technique non générative, rendant la détection plus complexe. Pour le dataset Text2Img, les modèles affichent de bonnes performances, montrant une meilleure capacité de généralisation pour les images générées par la même technique que le dataset d'entraînement. Enfin, sur le dataset GAN, les performances chutent de manière significative pour les deux modèles, suggérant des difficultés à généraliser aux images générées par des techniques différentes comme les GAN.

5.3 Approche Spectre de Fourier

Dataset	Accuracy	Precision	Recall	F1-Score	Auc
Dataset inpainting	92.05	98.06	85.91	91.58	99.12
Dataset Text2Image	63.75	59.87	83.07	69.58	71.97
Dataset Insight	71.30	66.30	87.83	79.03	81.15
Dataset Gan	65.67	61.65	85.26	71.55	72.02

Table 5.5: Performances de l'approche de Fourier sur le dataset Inpainting

Le tableau 5.5 révèle des différences notables entre les datasets. Le modèle montre une excellente performance sur le dataset Inpainting, indiquant une forte capacité de détection pour ce type de deepfakes grâce à un entraînement spécifique. Cependant, les performances diminuent significativement sur les autres datasets. Cette diminution suggère que le modèle, bien qu'efficace pour les deepfakes d'Inpainting, généralise moins bien pour les deepfakes générés par d'autres méthodes.

5.4 Approche Réduction de dimensions

Dataset	Accuracy	Precision	Recall	F1-Score	Auc
Dataset inpainting	64.05	64.63	64.23	64.42	69.74
Dataset Text2Image	72.26	26.75	35.33	30.44	40.24
Dataset Insight	36.39	14.62	33.32	20.32	24.53
Dataset Gan	48.80	22.53	39.43	28.67	32.95

TABLE 5.6: Performances de l'approche Autoencoeur sur le dataset Inpainting

Les résultats du modèle de détection de deepfakes basé sur l'autoencodeur montrent une performance globalement médiocre. L'accuracy et le F1-score sur le dataset d'entraînement (Inpainting) sont modérément faibles, indiquant une capacité limitée du modèle à capturer des caractéristiques discriminantes pertinentes. Lorsqu'évalué sur les autres datasets, les performances chutent de manière significative. Sur le dataset Text2Image, bien que l'accuracy soit légèrement meilleure, la précision et le rappel sont extrêmement faibles, reflétant un grand nombre de faux positifs et de faux négatifs. Les résultats sur les datasets Insight et GAN sont particulièrement mauvais, avec des métriques de performance indiquant une incapacité presque totale à différencier les images réelles des deepfakes. Cela suggère que l'autoencodeur[18] n'a pas réussi à généraliser au-delà des données d'entraînement et qu'il manque d'efficacité pour capturer des traits distinctifs des deepfakes générés par d'autres techniques.

5.5 Conclusion

Dans ce projet, nous avons exploré deux principales approches pour la détection de deepfakes : une approche directe basée sur des modèles CNN et une approche fréquentielle basée sur le spectre de Fourier.

L'approche directe (CNN) a montré une bonne capacité à détecter les deepfakes générés par Stable Diffusion, particulièrement pour les images Text2Image. Cependant, elle a rencontré des difficultés à généraliser cette détection à d'autres générateurs de deepfakes tels que InsightFace et GAN.

L'approche basée sur le spectre de Fourier a confirmé notre hypothèse initiale concernant la présence d'artefacts de diffusion dans les images générées, démontrant ainsi son efficacité pour la détection de deepfakes.

En revanche, l'approche basée sur la réduction de dimensionnalité avec autoencodeurs a été moins concluante, probablement en raison de la difficulté à extraire des caractéristiques significatives des visages réels.

En raison de contraintes techniques de dernière minute, nous n'avons pas pu évaluer nos approches sur les autres datasets dans le temps imparti. Néanmoins, nos résultats indiquent que les méthodes basées sur les CNN et les spectres de Fourier sont prometteuses pour la détection de deepfakes générés par Stable Diffusion, bien que des améliorations soient nécessaires pour une meilleure généralisation à d'autres types de générateurs de deepfakes.

Conclusion Générale

Ce projet visait à développer des approches robustes pour la détection de deepfakes, en se concentrant particulièrement sur les images générées par des modèles comme Stable Diffusion. À travers une exploration approfondie, nous avons testé et comparé trois méthodes principales : une approche directe basée sur des modèles CNN, une approche fréquentielle utilisant le spectre de Fourier, et une approche de réduction de dimensionnalité avec des autoencodeurs.

L'approche directe basée sur des CNN a montré une bonne capacité à détecter les deepfakes générés par Stable Diffusion, en particulier pour les images Text2Image. Cependant, elle a rencontré des difficultés à généraliser cette détection à d'autres générateurs de deepfakes tels qu'InsightFace et GAN. Cette limitation est attribuable aux différences dans les techniques de génération, InsightFace utilisant le face swapping, une méthode non générative, ce qui complique la détection.

L'approche fréquentielle, utilisant le spectre de Fourier, a confirmé notre hypothèse selon laquelle les artefacts spécifiques de la diffusion peuvent être efficacement capturés dans le spectre de Fourier. Cette méthode a montré une excellente performance sur le dataset d'entraînement mais a eu du mal à généraliser à d'autres types de générateurs de deepfakes.

En revanche, l'approche de réduction de dimensionnalité avec des autoencodeurs s'est avérée moins efficace. Les résultats ont montré une capacité limitée à extraire des caractéristiques significatives des visages réels, ce qui a conduit à des performances médiocres sur les datasets de test. Cette méthode n'a pas réussi à généraliser au-delà des données d'entraînement, en particulier pour les deepfakes générés par des techniques différentes.

Pour améliorer les performances futures, plusieurs perspectives se dessinent. D'abord, la diversification des sources de données truquées est essentielle pour améliorer la généralisation des modèles. Ensuite, la fusion des approches fréquentielle et directe pourrait conduire à des modèles plus robustes, capables de capturer des caractéristiques à la fois spatiales et fréquentielles des images. Enfin, l'utilisation de modèles de réduction de dimensionnalité plus efficaces, tels que FaceNet, pourrait améliorer la capacité à distinguer les images réelles des deepfakes.

En conclusion, bien que certaines approches aient montré des résultats prometteurs, il est évident qu'aucune méthode unique ne peut répondre à tous les défis posés par la détection de deepfakes. Des améliorations et des recherches supplémentaires sont nécessaires pour développer des solutions plus robustes et généralisables. Nos travaux fournissent une base solide pour de futures investigations et améliorations dans le domaine de la détection de deepfakes, contribuant ainsi à la lutte contre la désinformation et à la protection de la confiance numérique.

Bibliographie

[1] Haixu Song et al. *Robustness and Generalizability of Deepfake Detection: A Study with Diffusion Models*. 2023. arXiv : 2309.02218 [cs.CV].

[2] Jia Guo et Jiankang Deng. *InsightFace: State-of-the-art 2D and 3D Face Analysis Project*. https://github.com/deepinsight/insightface. Accessed: 2024-05-25. 2024.

[3] *CelebFaces Attributes (CelebA) Dataset*. https://www.kaggle.com/datasets/jessicali9530/celeba-dataset.

[4] BOJAN TUNGUZ. *1 Million Fake Faces*. https://www.kaggle.com/datasets/tunguz/1-million-fake-faces.

[5] *140k Real and Fake Faces*. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces.

[6] Ian J Goodfellow et al. « Generative Adversarial Nets ». In : *arXiv preprint arXiv:1406.2661* (2014). URL : https://arxiv.org/pdf/1406.2661.

[7] Diederik P Kingma et Max Welling. « Auto-Encoding Variational Bayes ». In : *arXiv preprint arXiv:1312.6114* (2013). URL : https://arxiv.org/pdf/1312.6114.

[8] Tero Karras, Samuli Laine et Timo Aila. « A Style-Based Generator Architecture for Generative Adversarial Networks ». In : *arXiv preprint arXiv:1812.04948* (2019). URL : https://arxiv.org/pdf/1812.04948.

[9] Robin Rombach et al. « High-Resolution Image Synthesis with Latent Diffusion Models ». In : *arXiv preprint arXiv:2112.10752* (2022). URL : https://arxiv.org/pdf/2112.10752.

[10] Karen Simonyan et Andrew Zisserman. « Very Deep Convolutional Networks for Large-Scale Image Recognition ». In : *arXiv preprint arXiv:1409.1556* (2014). URL : https://arxiv.org/pdf/1409.1556.

[11] Great Learning. *Everything You Need to Know About VGG16*. https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918. Accessed: 2024-05-25. 2024.

[12] Gao Huang et al. « Densely Connected Convolutional Networks ». In : *arXiv preprint arXiv:1608.06993* (2016). URL : https://arxiv.org/pdf/1608.06993.

[13] *What is Python?* https://www.python.org/doc/essays/blurb/.

[14] *Tensorflow*. www.tensorflow.org.

[15] *Keras*. https://keras.io/about/.

[16] *OpenCV*. https://opencv.org/about/.

[17] F. Pedregosa et al. « Scikit-learn: Machine Learning in Python ». In : *Journal of Machine Learning Research* 12 (2011), p. 2825-2830.

[18] Samy Aghiles Aouabed. *Image translation*. https://github.com/Samy-Abd/Image-translation. Accessed: 2024-05-25. 2024.

[19] Kaipeng ZHANG et al. « Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks ». In : *IEEE Signal Processing Letters* 23.10 (2016), p. 1499-1503. DOI : 10.1109/LSP.2016.2603342.