Projet de Recherche (PRe)

Spécialité : STIC/LeCS
Année scolaire : 2023/2024

Détection et reconnaissance de visage dans des images d'archive

NON CONFIDENTIEL

Auteur : Thomas Varin **Promotion : 2025**

Tuteur ENSTA Paris : François Goulette
Tuteur organisme d'accueil : Tanguy Gernot

Stage effectué du 03 / 06 / 2024 au 23 / 08 / 2024

NOM de l'organisme d'accueil : Laboratoire GREYC
Adresse : Bâtiment F, ENSICAEN, Boulevard Maréchal Juin, 14020 Caen

1 Résumé

Ce rapport de stage présente une comparaison de plusieurs outils de détection et de reconnaissance de visages dans des images. A l'aide de certaines métriques telles que le f-score ou encore l'EER, il est possible de mesurer les performances de ces derniers.
Ce rapport se concentre sur les performances des modèles qui sont face_recognition et VGGFace avec MTCNN, notamment au travers de leurs implémentations les plus populaires en Python.
En plus de la comparaison sera proposé une implémentation d'un outil qui permet de filtrer des images, en fonction de si ces dernières contiennent ou non le visage d'une personne (en fournissant une ou plusieurs références de ce dernier).

Mots-clés :
Reconnaissance/Détection de visage, Outil, Comparaison, Performances

2 Abstract

This internship report presents a comparision between several face detection and face recognition algorithms. Helped by standard metrics such as f-score or EER, it is possible to measure the performances of these very algorithms.
This report focuses on the two algorithms that are face_recognition and VGGFace with MTCNN, mostly by using their most known implementations in Python.
In addition to this, we also provide the implementation of a tool which allows one to filter images, whether or not they contain the face of a specific person (using one or many references of this face).

Keywords :
Face detection/recognition, Tool, Comparision, Performances

Table des matières

3 Introduction

La détection et la reconnaissance de visages sont récemment devenues des technologies très largement utilisées dans des domaines tels que la sécurité ou encore la biométrie. Ces systèmes utilisent des algorithmes capables de détecter et de reconnaître des personnes dans des images. Cependant les performances peuvent grandement varier en fonction du modèle utilisé.

Le présent rapport se concentre sur la comparaison d'outils de détection et de reconnaissance de visages. Plus précisément, deux implémentations seront étudiées : face_recognition et VGGFace associé à MTCNN. À travers une évaluation basée sur des métriques telles que le f-score et le taux d'erreur égale (EER), l'objectif est de comparer l'efficacité de ces deux modèles.

Enfin, l'implémentation d'un outil sera proposée. Cet outil permet de filtrer des images en fonction de la présence ou non d'un visage, en se basant sur une ou plusieurs références fournies au préalable. Cette approche a pour but de faciliter l'utilisation de la reconnaissance faciale dans divers cas d'usage, notamment dans le cadre de l'archive numérique.

En plus de cela, ce rapport fait part de la contribution apportée pour un projet réalisé durant un événement organisé par le laboratoire : la Cyber School in Normandy. Ce projet, proposé par la gendarmerie nationale, consistait en l'étude des techniques de création et de détection de vidéos truquées par l'intelligence artificielle. Pour ma part, je me suis retrouvé à travailler sur un projet à la demande de la gendarmerie nationale.

4 Projet pour la Cyber School in Normandy

4.1 Introduction

La Cyber School in Normandy est un événement organisé par l'école ENSICAEN et le laboratoire GREYC. Pour sa première édition, elle s'est tenue du 24 juin au 5 juillet à Caen. Durant une dizaine de jours, des étudiants du monde entier se sont réunis pour assister à de conférences faites par des experts internationaux, ainsi que pour travailler ensemble sur un projet de recherche.

4.2 Projet de création et de détection de deep-fake pour la gendarmerie nationale

4.2.1 Contexte du projet

Dans un contexte de plus en plus tourné vers la visio-conférence, de nouvelles méthodes d'attaques ont vu le jour. Là où l'arnaque à l'appel téléphonique commençait à s'estomper, notamment grâce à de nombreuses méthodes de prévention, l'arrivée de l'intelligence artificielle à rebattu les cartes.

Cette technologie, toujours en progression, connaît des dérives de plus en plus dangereuses. Les attaquants l'utilisent maintenant dans l'optique de tromper les employés de certaines entreprises, et plus spécifiquement les personnes qui ont accès à des ressources de cette dernière.

Alors qu'il y a quelques années, les arnaques téléphoniques ne nécessitaient que de l'ingénierie sociale, les employés sont maintenant avertis et ont certains protocoles à respecter. Cependant, de nouveaux cas de l'attaque que l'on nomme "l'attaque au président" ont été recensés, comme le rapporte le journal Ouest-France[8]. Dans cet article du 4 février 2024, le journal rapporte l'usage de l'IA pour l'attaque au président.

Dans cette histoire, les attaquants se sont fait passer pour des membres du conseil d'administration de l'entreprise, comportant notamment le directeur financier. D'après un haut responsable de la police, "la vidéo-conférence impliquait plusieurs participants, mais que tous, à l'exception de la victime, étaient des fakes".

La demande de la gendarmerie était de faire un travail de recherche et de documentation sur les techniques de création et de détection du deep-fake (ou hyper-trucages en français). Cette section est fortement inspirée du rendu que nous avons fait et qui se trouve en annexe.

4.2.2 Techniques de création du deep-fake

La création de deep-fake peut se faire de plusieurs manières différentes. Dans notre situation, nous sommes intéressés par une branche plus précise du deep-fake : nous voulons animer le visage d'une personne existante. Pour ce faire, il existe deux méthodes générales.

La première, qui est la plus complexe à mettre en place, consiste à prendre une vidéo et à modifier le mouvement du visage, notamment avec l'utilisation de GAN (Generative Adversarial Networks). Cette technique, bien que fonctionnant assez bien sur des images fixes, ne semble pas convenir sur des visage animés, que cela soit sur une vidéo ou encore pire sur un flux en direct.

La seconde méthode, et celle que nous avons employé, s'appelle le face swapping (ou échange de visages en français). Cette méthode implique d'avoir une vidéo source et une vidéo cible, et se décompose en 6 étapes :

Premièrement, le modèle doit détecter et extraire les visages présents dans les deux vidéos. Ensuite, il faut réaliser une étape de localisation des points clés du visage, tels que les yeux, la bouche ou encore le nez. Après cela, on procède à ce que l'on appelle l'extraction de caractéristiques. Comme expliqué dans la deuxième partie de ce rapport, les caractéristiques (ou features) sont un grand vecteur de nombres entre 0 et 1 qui caractérisent un visage d'après le modèle qui les extrait. Une fois ces features extraites, l'algorithme va les échanger pour pouvoir rendre une vidéo qui met le visage de la source à la place du visage de la cible. Enfin, la plupart du temps il est nécessaire de traiter cette vidéo manuellement pour corriger quelques défauts, notamment dûs à la différence de teint ou d'accessoires entre les deux personnes échangées.

Étant limités par les deux semaines de la Summer School, nous avons trouvé le dépot GitHub [11], qui semblait bien fonctionner. Après de longue périodes de résolutions de problèmes, notamment avec les cartes graphiques qui ne voulaient pas fonctionner et l'impossibilité de trouver des explications claires pour l'utilisation sur un système Linux, nous avons adapté le tutoriel vidéo [12] pour le faire fonctionner sous Linux. En modifiant quelques scripts, nous avons réussi à faire fonctionner l'outil, afin de produire une vidéo plutôt convaincante.

4.2.3 Techniques de détection du deep-fake

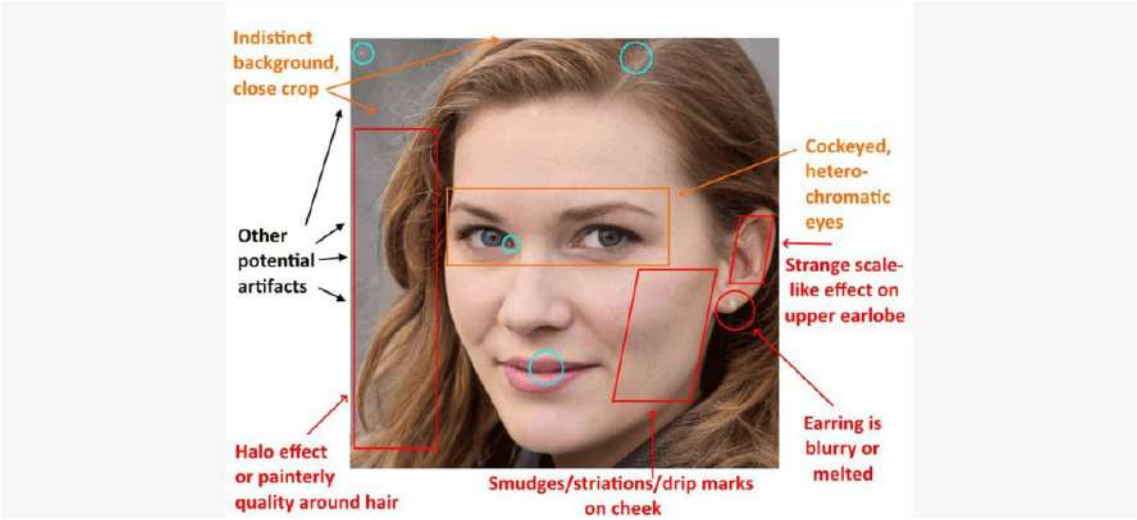
4.2.3.a Techniques basées sur l'image

Dans la plupart des vidéos qui comportent du deep-fake, il est possible de retrouver des erreurs qui ne sont pas détectables par un humain. Par exemple, on peut utiliser l'outil en ligne [21], qui nous permet d'analyser et de détecter des trucages dans des images.

(a) Image truquée (b) Détection d'erreurs de Forensically

FIGURE 1 – Analyse d'image truquée avec Forensically

Encore, il est possible de voir directement quelques artefacts à l'oeil nu, ou bien de les détecter avec un logiciel.



(a) Crédit : AP Photo

FIGURE 2 – Artefacts sur une image truquées

Le problème que nous pouvons rencontrer avec ce type d'outil d'analyse d'image est que parfois, dans des vidéos truquées, toutes les images sont bien faites et "indétectables" mais l'ensemble n'est pas cohérent. Il faut alors réaliser une détection qui traite directement le flux vidéo dans son ensemble. Une détection de ce genre est décrite dans les articles [18] et [6] et donne de très bon résultats.

4.2.3.b Techniques basées sur le comportement

L'idée des techniques basées sur le comportement est très simple et exploite les difficultés que peuvent rencontrer les attaquants lors d'une communication en direct. Ici, l'objectif est de "casser" la chaîne de production du deepfake présenté précédemment, ou au moins d'en détecter quelques faiblesses.

Déjà, une chose à noter est que dans beaucoup de cas de deep-fake, les personnes simulées ne clignent pas des yeux, ou au moins de façon étrange. Cet aspect est étudié dans [13] et présente des résultats assez impressionnants.

Pour en revenir à la "compromission" de la chaîne de production du deep-fake, des méthodes assez ingénieuse existe. En exploitant la communication en direct, il est possible de fausser la détection du visage cible, ou encore l'extraction de ses features, notamment en demandant à la personne dont on veut vérifier l'identité de faire certaines actions spécifiques. Comme spécifié dans [24], il est possible de rendre le visage cible indétectable, notamment en interagissant avec des objets pour masquer tout ou partie du visage, ou encore de tourner la tête de façon à ce que les caractéristiques du visage ne soit plus aussi facilement extraite. Cela aurait alors pour effet de faire croire à l'algorithme de deep-fake qu'il n'y a pas de visage à truquer, et donc de montrer directement le flux entrant, sans modification. Enfin, toujours dans l'optique de rendre la création du deep-fake beaucoup plus difficile, il est possible de faire beaucoup varier la luminosité de l'image (en demandant par exemple à la personne d'allumer une lampe et d'éclairer son visage). Cela aurait pour but de faire varier son teint, afin qu'il ne soit plus uniforme et que l'algorithme de deep-fake ne puisse pas générer d'images avec un teint très hétérogène (cela serait dû au fait qu'il n'y ait que très peu voire pas du tout d'image de ce type dans les données de la source)

5 Détection et reconnaissance de visage dans des images

5.1 Introduction

Dans un monde de plus en plus numérique, la collecte et le traitement d'archives d'auteurs posent de nouvelles difficultés. Ce sont ces difficultés que rencontre l'Institut Mémoires de l'Édition Contemporaine (IMEC). Le rôle de cet institut est de collecter et traiter les archives d'auteurs contemporains qui peuvent être conservées sur des supports divers et variés. On retrouve un nombre faramineux de types de données différents. Du simple fichier texte au fichier de versionnage en passant par la configuration de l'éditeur de texte, beaucoup de fichier doivent être traités.

En plus de ces fichiers qui concernent directement la personne en tant qu'artiste, il n'est pas rare de trouver d'autres contenus plus personnels, tels que des mails ou encore des photos. En effet, il est très courant pour les artistes contemporains d'utiliser leurs appareils personnels pour leurs oeuvres.

Dans le cadre de l'IMEC, l'objectif est de mettre à disposition des chercheurs les informations qui sont retrouvées dans les archives. Cependant, une question se doit d'être posée : "Tous les documents sont-ils communicables ?"

Dans certains cas tels que les différentes version d'une oeuvre, la réponse est très souvent positive. Malheureusement, certains cas sont plus litigieux et nécessitent un accord préalable des ayant-droits, en passant d'abord par le traitement manuel d'un archiviste. Pour le moment, les outils d'aide aux traitement des archives sont plutôt limités, et plus particulièrement dans le traitement des images.

5.1.1 Environnement de travail

Comme expliqué précédemment, les archives numériques se présentent sous différentes formes, de la simple disquette à l'ordinateur complet, tout ce qui contient de la donnée peut être utile pour retracer la vie d'un artiste ou d'une de ses oeuvres.

Cette diversité est intéressante mais apporte une contrainte sévère, il faut pouvoir lire tous les supports, certains étant particulièrement anciens. Alors, dans un but de simplicité, certains outils de Forensique ont été développés afin d'aider à la lecture de ces supports.

Un exemple plutôt connu est le logiciel Autopsy. Basé sur la librairie The Sleuth Kit et sur PhotoRec, ce logiciel permet d'analyser des images disques et d'y appliquer des filtres. De manière générale, les logiciels tels qu'Autopsy, ou encore celui en cours de développement par le GREYC G'DIP, utilisent pour la plupart le système de filtres.

Un filtre peut être vu comme une simple fonction, qui prend en entrée un ensemble de fichiers et qui en renvoie un sous-ensemble. On peut citer par exemple un filtre assez commun d'Autopsy, celui qui consiste à récupérer l'ensemble des fichiers qui contiennent une adresse e-mail.

Tous ces filtres peuvent être combinés au moyen d'opérateurs ensemblistes assez classiques, tels que des intersections, des unions...

Dans certains cas, certains filtres peuvent donner un score. Il est parfois utile de donner avec un certain taux de fiabilité certaines informations. Par exemple, certains filtres d'Autopsy permettent d'estimer la probabilité qu'un fichier ait été chiffré, notamment en calculant son entropie. Ensuite, il est très facile avec un seuil de ne prendre que les fichiers avec un bon score pour retomber sur un ensemble de fichiers, et retrouver les opérateurs ensemblistes.

5.1.2 Motivations de ce stage

Afin d'aider le travail des archivistes, notamment dans la gestion de la communicabilité des images, il existe déjà quelques filtres qui peuvent détecter des visages. Cependant pour le moment, aucun n'est capable de reconnaître l'appartenance du visage.

Dans le cadre de l'archivage numérique, on peut retrouver un nombre important de photos personnelles qu'il serait important de pouvoir filtrer. Pour ce faire, l'idée serait de diviser ce filtre complexe en plusieurs filtres plus élémentaires que l'on pourrait habilement combiner.

Pour ce stage, nous nous intéresserons à un filtre capable de détecter et reconnaître les visages, notamment à l'aide d'une ou plusieurs références d'une personne.

L'intérêt de ce filtre est de pouvoir faire un pré-filtrage des photos qui pourrait contenir le visage de certains membres de la famille de l'auteur. Le but n'est pas de remplacer le traitement manuel, mais au moins de pouvoir indiquer ces visages pour accélérer le traitement.

De plus, une idée a été suggérée pendant une présentation de l'outil par une personne de l'IMEC. Il serait possible de stocker les références de plusieurs artistes déjà rencontrés, afin de faire des liens entre les différentes archives et pouvoir lier deux artistes entre eux.

5.2 État de l'art

Il existe déjà quelques outils dont la technologie permettrait de réaliser notre tâche. On peut citer par exemple PimEyes[16], TinEye[19] ou encore la fonction de recherche inversée Yandex [23].

Dans un usage encore plus proche du notre, le système d'exploitation de téléphone iOS, à partir de sa version 14, permet de classifier des images en fonction des personnes apparaissant dans ces dernières. Après avoir demandé l'association visage/nom, le système est capable de regrouper toutes les photos contenant la dite personne. D'un autre côté, le géant de la tech Google propose une fonctionnalité similaire, avec son application Google Photos. Ces deux technologies permettent de faire de la classification d'images en fonction des visage présents, ce qui est aussi notre objectif final.

Le problème ici est que nous faisons face à des logiciels/outils propriétaires et dont on ne peut pas trouver le code source. Alors, il est nécessaire de créer notre propre outil, qui plus est dans un langage qui peut permettre de créer des filtres dans Autopsy/G'DIP comme Python par exemple.

5.3 Comparaison des algorithmes et choix des paramètres

Pour mettre en place ce filtre, il est nécessaire de trouver une solution qui détecte et extrait les caractéristiques (que nous appellerons par la suite features) des visages. Ensuite, une fois la fonction de distance choisie, il faut trouver un seuil en deçà duquel on considère que deux visages appartiennent à la même personne. Voyons ces deux points plus en détail.

5.3.1 Détection et Extraction

5.3.1.a Description des algorithmes

Pour ce qui est de la détection et l'extraction de visages, deux choix semblent s'offrir à nous.

Le premier est une bibliothèque python "2-en-1" qui s'appelle face_recognition [3]. Cette bibliothèque est capable de beaucoup de choses, notamment localiser et comparer des visages, extraire des features ou encore afficher des points-clés du visage tels que la bouche, le nez ou encore les yeux.

Le second sépare bien la partie détection de la partie reconnaissance. En effet, la partie détection est faite par un bibliothèque qui s'appelle MTCNN [10]. Cette dernière implémente l'article [25], qui utilise les Multi-task Cascaded Convolutional Networks (MTCNN). Une fois le visage extrait, on le passe à un autre réseau de neurones, VGGFace, qui se base sur les deux articles [15, 5].

Par la suite, et sauf cas contraire explicite, lorsque nous nous référerons à face_recogntion, MTCNN ou VGGFace, nous parlerons des bibliothèques et non des articles/modèles théoriques.

Malgré des designs complètement différents, face_recognition et VGGFace fournissent tous les deux une liste de features. Ces dernières ne semblent pas forcément rattachées à des traits classiques (écartement des yeux, rapport entre plusieurs angles du visage, ...) mais plutôt à une liste de nombres "arbitraires" qui sont la sortie d'un réseau de neurones. Dans le cas de face_recognition, on se retrouve avec 128 features, là où VGGFace nous en donne 512. Ces features permettent de "décrire" les visages trouvés, et plus les visages sont proches, plus on s'attend à ce que leurs features soit "proches" (cette définition dépend de la distance choisie)

Pour ce qui est de la distance justement, on retrouve assez classiquement la distance euclidienne pour face_recognition (dans l'implémentation directement) et plutôt la distance cosinus [14] pour VGGFace dans un des tutoriels proposés directement sur le GitHub de VGGFace [4] [17].

5.3.1.b Éléments de mesure des performances

Afin de faire une bonne comparaison de performances, il nous faut des outils statistiques pertinents. Dans notre contexte, on définit les grandeurs suivantes (on notera T_+ le nombre de visages correctement acceptés, F_+ le nombre de visages faussement acceptés, T_- le nombre de visages correctement rejetés, F_- le nombre de visages faussement rejetés) :

On notera FAR le taux de fausse acceptation, que l'on définit par :

$$FAR(t) = \begin{cases} \frac{F_+(t)}{F_+(t)+T_-(t)}, & \text{si } F_+(t) \neq 0 \\ 0, & \text{sinon} \end{cases} \tag{1}$$

On notera FRR le taux de faux rejet, que l'on définit par :

$$FRR(t) = \begin{cases} \frac{F_-(t)}{F_-(t)+T_+(t)}, & \text{si } F_-(t) \neq 0 \\ 1, & \text{sinon} \end{cases} \tag{2}$$

On notera EER le taux d'erreur égal, il se situe à l'ordonnée du point d'intersection entre la courbe de FAR et la courbe de FRR. On le définit par :

$$EER = \min_{0 \leq t \leq 1} \max\{FAR(t), FRR(t)\} \tag{3}$$

On notera enfin F_1 le f-score, que l'on définit par :

$$F_1 = \frac{2T_+(t)}{2T_+(t) + F_+(t) + F_-(t)} \tag{4}$$

5.3.2 Banque d'images utilisées pour les mesures de performances

Afin de faire une comparaison fiable, qui ne dépende pas du dataset, il convient d'en trouver un qui ne fasse pas partie des données d'entraînement. La base de données d'entraînement de VGGFace est publique et directement donnée dans [5]. Pour face_recognition, la tâche se complique un petit peu. En effet, le créateur de cette librairie très répandue ne semble pas vouloir révéler la base de données utilisée. Cependant, il précise "The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark" dans son README.md [3].

Alors, on peut en déduire que ce dataset n'est pas utilisé pour l'entraînement de face_recognition.

Labeled Faces in the Wild [9] est un dataset public créé par Gary B. Huang, Manu Ramesh, Tamara Berg et Erik Learned-Miller. Ce dernier contient un ensemble de photos labélisées avec une seule personne par photo. Au total, cette base de donnée contient 13233 images, 5749 personnes, et 1680 personnes avec au moins deux images. Puisque nous avons besoin d'au moins une référence par personne, nous ne nous intéresserons qu'à ces fameuses 1680 personnes.

De plus, un autre dataset qui ne semble pas être utilisé par les deux algorithmes est "Portrait and 26 photos" [20]. Disponible en version payante, cette banque d'images propose une version d'essai, avec 14 dossiers qui contiennent le portrait d'une personne et 26 photos de leur quotidien (la base complète contient 272 dossiers). Cette banque d'images nous permettra de mettre en évidence un problème de face_recognition un petit peu plus tard.

5.3.3 Problème du remplissage de l'espace

Un problème qui se pose est le problème du remplissage de l'espace. Pour simplifier l'explication, nous supposerons que nos features sont toutes entre 0 et 1 (ce qui est très proche de la réalité). De plus, on supposera que l'on utilise la distance euclidienne. Enfin, nous supposerons que les features sont uniformément distribuées. En notant d le nombre de features données par notre réseau de neurones, cela revient à dire que notre réseau prend en entrée une image et renvoie un (ou plusieurs) point dans l'espace métrique $[0,1]^d$.

Vérifier si la distance entre un point et une référence (un autre point) est plus petite que t est équivalent à vérifier si ce point appartient à la boule de rayon t et de centre la référence.

Le volume d'une hyper-boule de rayon t en dimension d vaut [7] :

$$V^{(d)}[t] = \frac{\pi^{d/2} t^d}{\Gamma(\frac{d}{2}+1)} \tag{5}$$

Avec cette formule, on trouve $V^{(128)}[t] = \frac{\pi^{64} t^{128}}{64!}$ et $V^{(512)}[t] = \frac{\pi^{256} t^{512}}{256!}$.

En prenant un trop grand nombre de références, il serait possible de "remplir" l'espace (pas forcément à 100% mais une proportion non négligeable). Par exemple, pour un threshold de 0.6 en dimension 128, l'aire de la boule est d'environ 2.10^{-86}. Donc avec environ 5.10^{85} boules, on pourrait remplir l'espace. Ce nombre paraît extraordinairement grand, mais se base sur beaucoup d'hypothèses simplificatrices qui rendent maximale le nombre de boules. On en connaît une borne supérieure, mais en pratique ce nombre est beaucoup plus bas, notamment sans l'hypothèse d'uniformité.

Ce qu'il faut retenir de cela est que plus augmente le nombre de références, plus un point aléatoire dans l'espace a de chance d'appartenir à une boule et donc d'être assigné à une référence, quand bien même les visages soient différents.

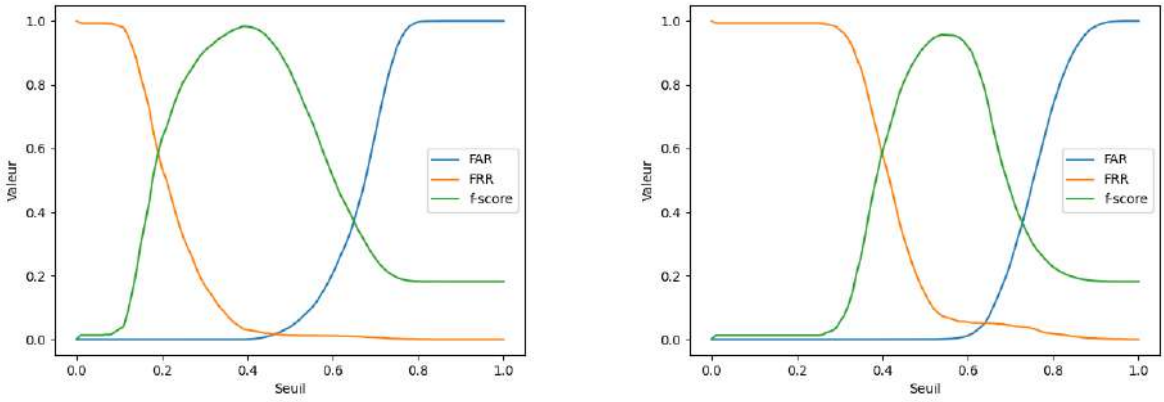
Dans notre cas, le nombre de classes est au plus 1680, ce qui permet de s'affranchir de ce genre de problème, mais il me semblait important de le mentionner.

5.3.4 Résultats obtenus

5.3.4.a Sur le dataset : Labelled Faces in the Wild

Pour LWF, nous avons trié les classes en fonction du nombre d'images de ces dernières. Pour des questions de performances et en essayant de prendre quand même en compte le phénomène précédant, nous aurons des résultats pour 10, 20, 50, 100, 200, 500, 1000, et 1680 classes.

Pour les 10 classes les plus peuplées, nous avons 1533 images :

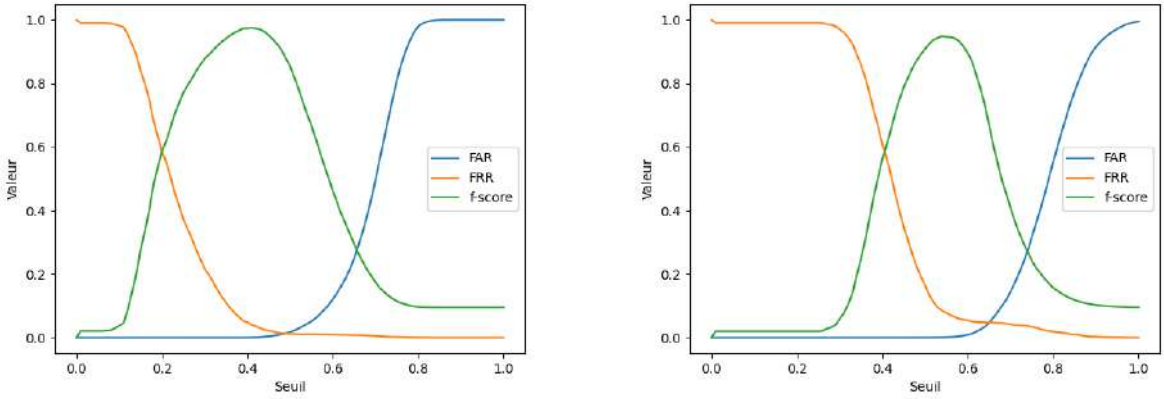


(a) VGGFace/MTCNN

(b) face_recognition

FIGURE 3 – FAR, FRR et F1 avec 10 classes (LFW)

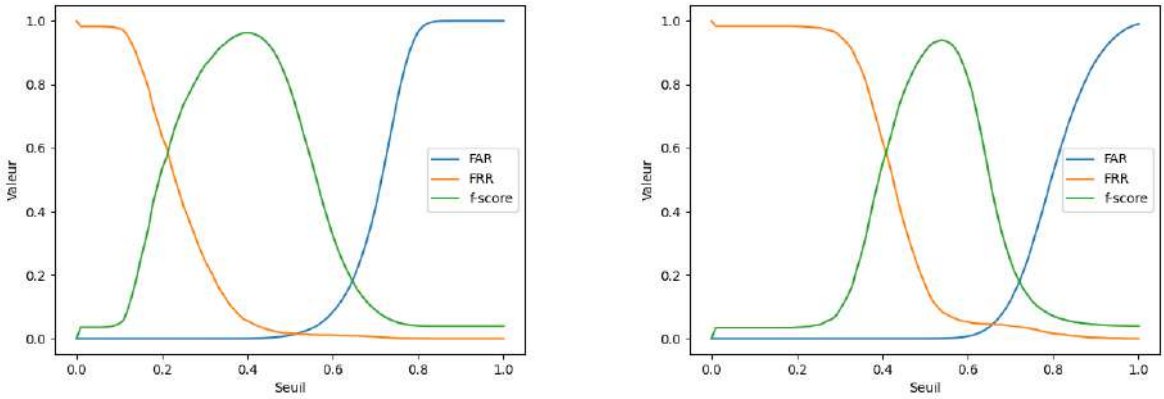
Pour les 20 classes les plus peuplées, nous avons 1906 images :



(a) VGGFace/MTCNN (b) face_recognition

FIGURE 4 – FAR, FRR et F1 avec 20 classes (LFW)

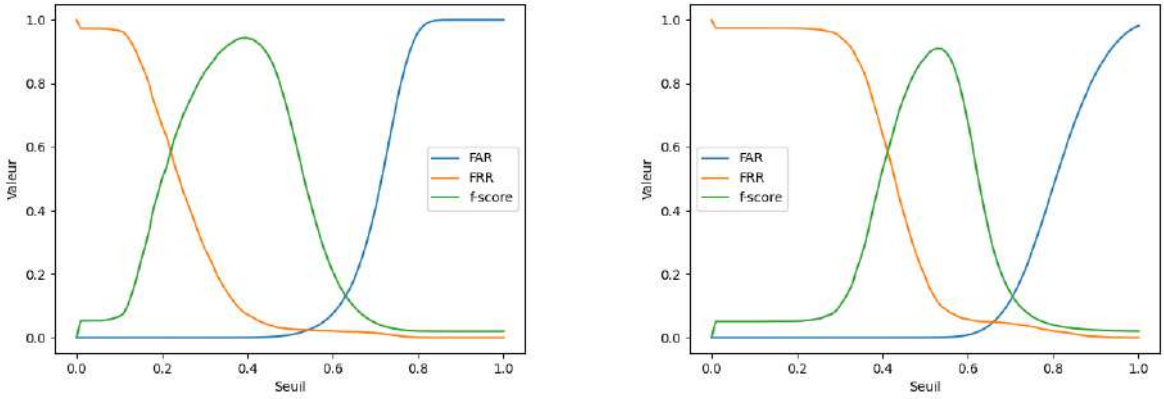
Pour les 50 classes les plus peuplées, nous avons 2773 images :



(a) VGGFace/MTCNN (b) face_recognition

FIGURE 5 – FAR, FRR et F1 avec 50 classes (LFW)

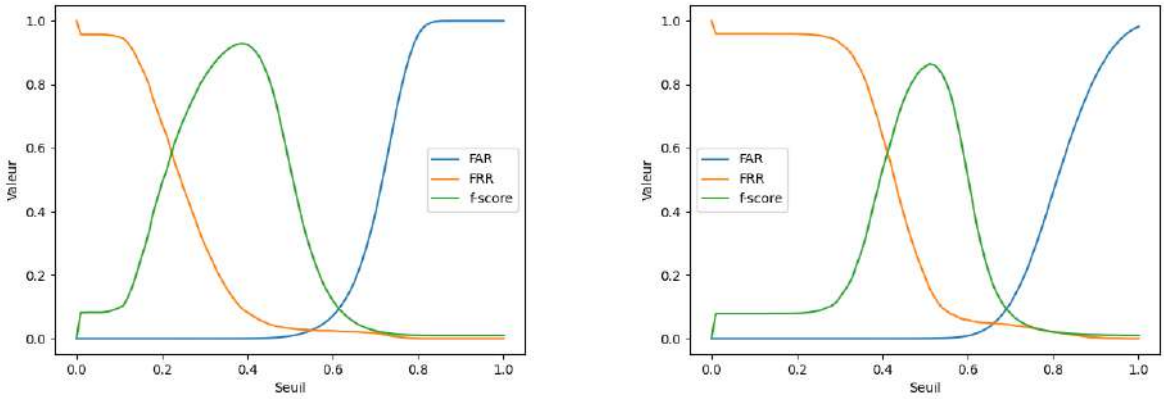
Pour les 100 classes les plus peuplées, nous avons 3651 images :



(a) VGGFace/MTCNN (b) face_recognition

FIGURE 6 – FAR, FRR et F1 avec 100 classes (LFW)

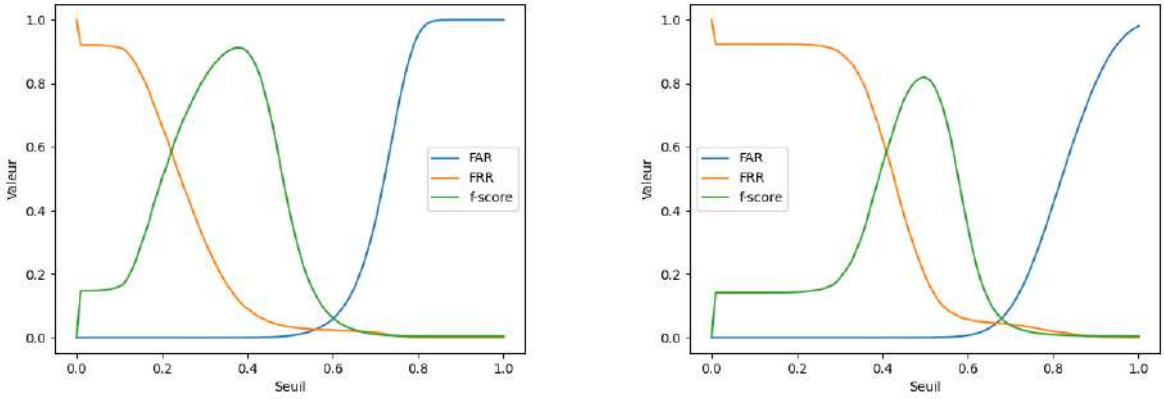
Pour les 200 classes les plus peuplées, nous avons 4686 images :



(a) VGGFace/MTCNN (b) face_recognition

FIGURE 7 – FAR, FRR et F1 avec 200 classes (LFW)

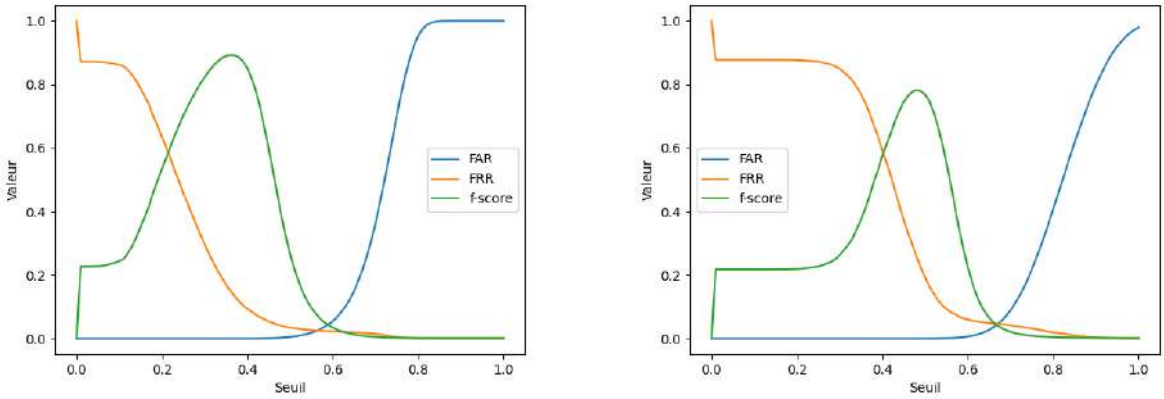
Pour les 500 classes les plus peuplées, nous avons 6293 images :



(a) VGGFace/MTCNN (b) face_recognition

FIGURE 8 – FAR, FRR et F1 avec 500 classes (LFW)

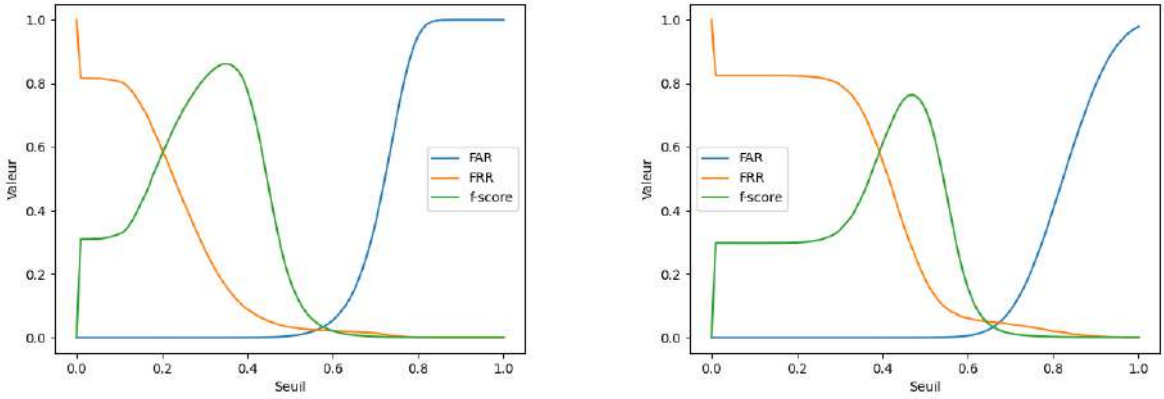
Pour les 1000 classes, nous avons 7804 images :



(a) VGGFace/MTCNN (b) face_recognition

FIGURE 9 – FAR, FRR et F1 avec 1000 classes (LFW)

Pour les 1680 classes, nous avons 9164 images :



(a) VGGFace/MTCNN (b) face_recognition

FIGURE 10 – FAR, FRR et F1 avec 1680 classes (LFW)

En reprenant 3, nous avons le tableau d'EER suivant :

Classes \ Algorithme	MTCNN/VGGFace	face_recognition
10	**0.017**	0.051
20	**0.013**	0.048
50	**0.015**	0.045
100	**0.023**	0.049
200	**0.026**	0.048
500	**0.025**	0.045
1000	**0.025**	0.046
1680	**0.024**	0.047

TABLE 1 – EER pour VGGFace et face_recognition avec LFW

On remarque ici que l'EER obtenu avec VGGFace et MTCNN est bien inférieur à celui obtenu par face_recognition. En effet, il est presque diminué de moitié dans chacun des tests. De plus, il arrive assez fréquemment que certains visages ne soient même pas trouvés par face_recognition. Sur les 9164 visages des 1680 classes de LFW, 35 n'ont pas été détectés.

Pour certaines images, la raison est compréhensible, comme un visage partiellement caché par un obstacle ou un accessoire :

(a) Caché par un obstacle (b) Caché par des accessoires

FIGURE 11 – Visages non détectés par face_recognition (LFW difficile)

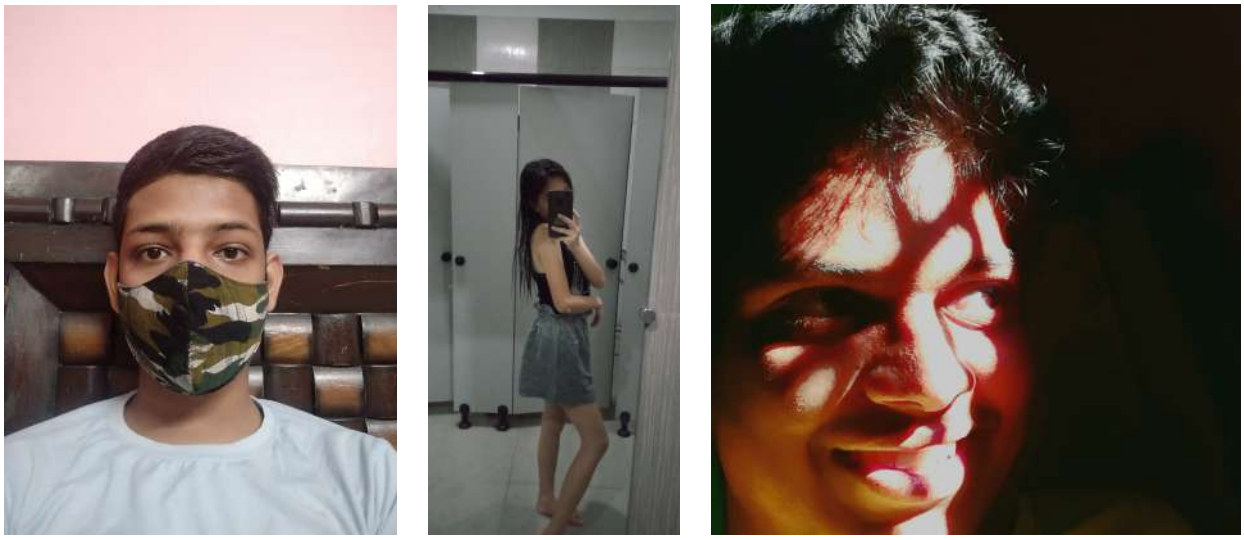
Pour autant, certaines images ne sont pas détectées alors que rien ne semble gêner la perception du visage :

FIGURE 12 – Visages non détectés par face_recognition (LFW facile)

5.3.4.b Sur le dataset : Portrait and 26 photos

Portrait and 26 photos [20] est une banque d'images contenant 272 dossiers contenant, pour chacun, un portrait et 26 photos du quotidien de cette personne. Cette banque en'est pas gratuite, mais une petite partie contenant 14 dossiers à été mise à disposition du grand public. Contrairement à LFW, qui connaît pas mal de biais statistiques (notamment au niveau du genre et de l'ethnie) que l'on peut voir dans [2], celle-ci (en tout cas la partie accessible gratuitement) ne comporte pas que des hommes caucasiens. En effet, on retrouve 5 femmes et 9 hommes, et seulement deux personnes caucasiennes.

En plus d'avoir une diversité plus importante, cette banque de données possède aussi une autre particularité : les images n'ont pas été recadrées et ne contiennent pas toujours qu'un visage. Cela permet alors de vérifier la capacité qu'ont les deux modèles à détecter les visages, ce que nous ne pouvions pas bien faire plus tôt. Enfin, certaines images sont "difficiles", pour les mêmes raisons que celles mentionnées avec LFW. En voici quelques exemples (les trois ne sont pas detectés par face_recognition) :



(a) Caché par un accessoire (b) Caché par un obstacle (c) Lumière très hétérogène

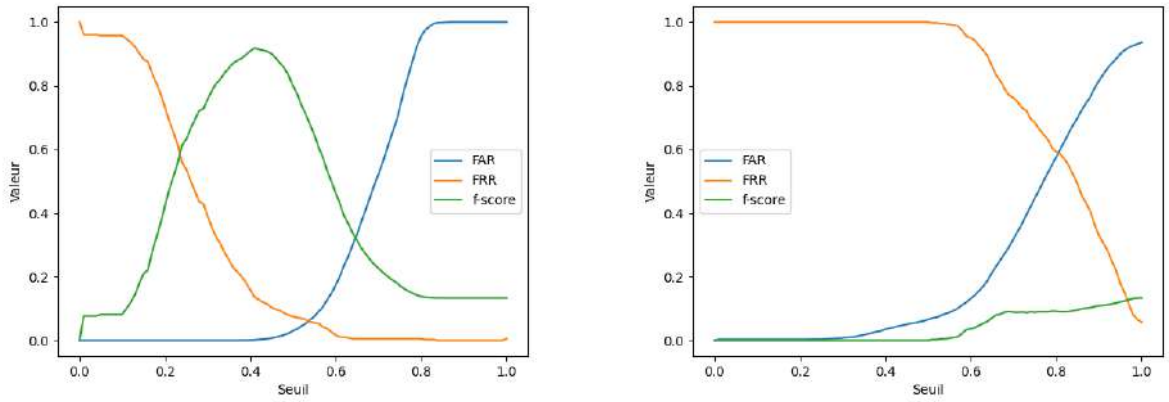
FIGURE 13 – Visages non détectés par face_recognition (Portrait difficile)

De la même manière que pour LFW, certaines images "simples" ne sont pas détectées non plus par face_recognition, comme par exemple :

FIGURE 14 – Visages non détectés par face_recognition (Portrait facile)

Au total, ce sont 19 images sur 378 qui ne possèdent aucun visage selon face_recognition, contre seulement deux pour MTCNN/VGGFace (en rappelant qu'il existe une image qui n'en contient aucun). D'ailleurs, on remarque que les images où face_recognition ne détecte pas de visage ne sont pas uniformément distribuées : certaines personnes sont toujours détectées, là où d'autres ne sont pas vues sur plusieurs photos (deux personnes ont 5 images considérées vides, et une autre personne en a 6).

Avec aussi peu d'images, il est inutile de traiter plusieurs nombres de classes. Nous avons alors les deux courbes et le tableau suivants :



(a) VGGFace/MTCNN

(b) face_recognition

FIGURE 15 – FAR, FRR et F1 avec 14 classes (Portrait and 26 photos)

Classes \ Algorithme	MTCNN/VGGFace	face_recognition
14	**0.061**	0.58

Table 2 – EER pour VGGFace et face_recognition avec Portrait

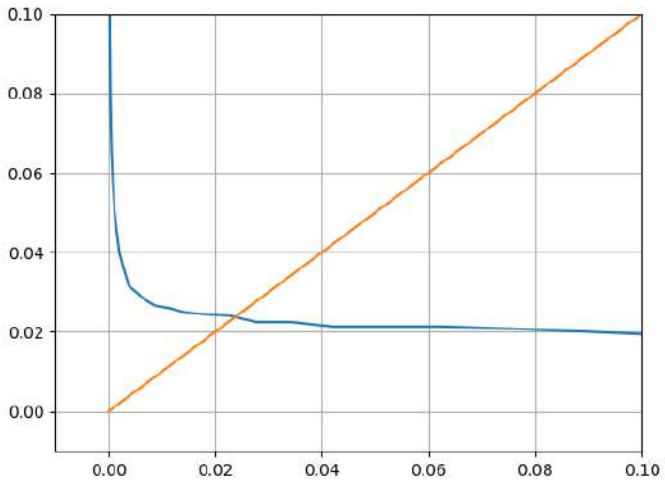
On voit tout de suite que VGGFace couplé à MTCNN est beaucoup plus efficace sur des photos réelles que face_recognition. En effet, les deux courbes ne se ressemblent même plus et les EER ont un facteur 10, encore une fois avec un meilleur résultat de VGGFace.

Nous pouvons donc en conclure que VGGFace (en tout cas dans notre usage de détection et reconnaissance de visages dans des images "complexes") semble plus efficace que face_recognition. Par la suite, nous utiliserons la bibliothèque VGGFace avec MTCNN.

5.3.5 Choix du seuil d'acceptation

Le choix du seuil est quelque chose d'assez difficile et arbitraire. En fonction de l'usage que nous en avons, il peut être plus intéressant de minimiser le FAR, le FRR ou encore l'EER. Comme expliqué dans [1] à la page 26, "Un seuil θ trop petit entraîne l'apparition d'un grand nombre de faux rejets, tandis qu'un seuil θ trop grand engendre un taux important de fausses acceptations". Il faut maintenant faire un choix entre la précision ou l'exhaustivité.

Comme proposé dans cette thèse, une chose qu'il est possible de faire est de tracer la courbe DET (pour Detection Error trade-off). L'idée est de tracer le FRR en fonction du FAR, les deux étant fonction du seuil. Ici, nous avons le résultat suivant (la courbe orange est la droite d'équation $y = x$) :



(a) VGGFace/MTCNN

Figure 16 – Courbe DET sur LFW pour 100 classes (VGGFace)

On retrouve bien notre EER qui vaut environ 0.24 mais on remarque quelque chose d'assez intéressant. En sacrifiant un petit peu de performances sur le FRR (en l'augmentant donc), il est possible de beaucoup réduire le FAR. En effet, voici quelques valeurs significatives que nous obtenons :

Seuil \ Mesure	FAR	FRR
0.542	0.024	0.024
0.520	0.014 (÷1.64)	0.024 (×1.07)
0.500	0.008 (÷2.70)	0.026 (×1.14)
0.480	0.005 (÷4.63)	0.030 (×1.29)
0.460	0.002 (÷8.47)	0.036 (×1.55)
0.440	0.0015 (÷15.55)	0.045 (×1.96)

TABLE 3 – Évolution des FAR et FRR en fonction du seuil choisi

On remarque alors quelque chose d'assez intéressant : il est possible de diviser notre FAR par 15.55 en multipliant notre FRR seulement par 1.96. Si l'on veut être un petit peu plus raisonnable, il est tout de même possible de diviser notre FAR par 4.6 en augmentant notre FRR de seulement 29%.

Dans le cas des archives numériques, il n'est pas vital d'avoir un FRR très faible, mais toujours utile (et probablement plus agréable) d'avoir un FAR faible. Nous prendrons alors pour la suite des seuils entre 0.5 et 0.46.

5.3.6 Gestion des références multiples

Lorsque nous avons affaire à de la reconnaissance faciale, il n'est pas rare d'avoir plusieurs références d'une même personne. Une question se pose alors : comment agglomérer les distances à nos multiples références ?

Intuitivement, nous aimerions une fonction qui puisse prendre en entrée un nombre arbitraire de valeurs et nous donner une unique distance (la distance d'un point à un ensemble fini). Nous aimerions que notre fonction de cumul de distances, que nous nommerons f, vérifie certaines propriétés élémentaires que voici :

Soit $n \in \mathbb{N}, (x_1, ..., x_n) \in \mathbb{R}^n, t \in \mathbb{R}, \sigma \in \mathfrak{S}_n$ et $x \in \mathbb{R}$

$$\min_{i=1,...n} x_i \leq f(x_1, ..., x_n) \leq \max_{i=1,...n} x_i$$

$$f(x, x, ..., x) = x$$

$$f(tx_1, ..., tx_n) = tf(x_1, ..., x_n)$$

$$f(x_{\sigma(1)}, ..., x_{\sigma(n)}) = f(x_1, ..., x_n)$$

$$x_i \mapsto f(x_1, ..., x_i, ..., x_n) \text{ est croissante}$$

Ces conditions sont assez spécifiques et représentent un ensemble de conditions nécessaires pour être une moyenne [22].

Alors cinq idées viennent assez naturellement :
— Le minimum
— Le maximum
— La moyenne arithmétique
— La moyenne géométrique
— La médiane

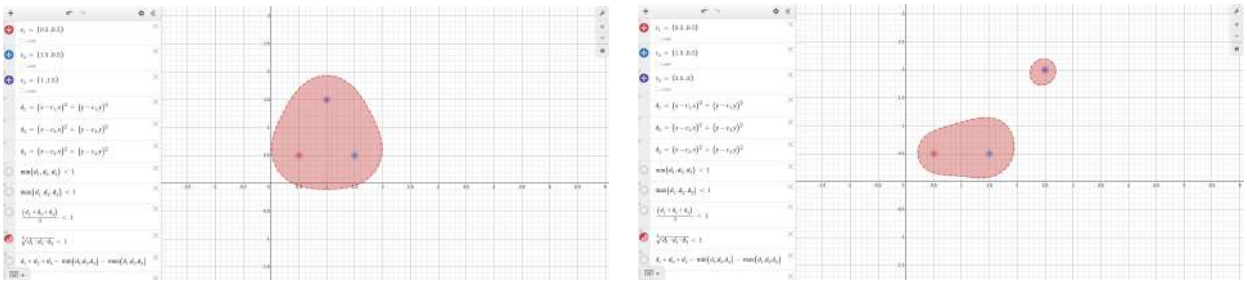
En utilisant le site internet Desmos, il est possible de se représenter ces cinq moyennes en deux dimensions. En construisant trois points et en calculant la distance de chaque point du plan avec ces points (en utilisant nos fonctions d'agglomération), on peut regarder l'ensemble des points qui sont à une distance inférieure à 1 dans le plan.

On remarque déjà un problème, lorsque les cercles sont trop éloignés, le maximum et la moyenne arithmétique produisent des ensembles vides. Cela indique que si une ou plusieurs références sont éloignées des autres (ce qui pourrait avoir pour origine une mauvaise manipulation ou une mauvaise détection par notre algorithme), aucune image n'est détectée. Comme nous voulons éviter ce genre de situations désagréables, nous pouvons dores et déjà supprimer ces deux fonctions.

Ensuite, une problématique du minimum est que si deux références sont proches, elle n'ont pas plus de poids que deux références éloignées l'une de l'autre. En effet, l'union de deux cercles aura toujours une aire plus faible que la somme des aires de ces deux cercles. Cela a pour effet de "renforcer" les anomalies, puisque deux anomalies couvrent une aire plus importante qu'une centaine de références proches.

Enfin, en ce qui concerne la médiane, le problème inverse se pose. Lorsqu'une référence est trop éloignée des autres, elle est complètement ignorée. Cela signifie qu'une image, pour être acceptée, a besoin d'être proche de $\frac{n}{2}$ valeurs mais peut être très éloignée des autres.

Il ne nous reste alors plus que la moyenne géométrique, qui ne semble pas avoir les problèmes cités précédemment. Lorsque l'on a deux références proches, on se retrouve avec une patatoïde qui est plus large du côté de ces deux dernières. De plus, lorsque l'on éloigne une référence des deux autres, cela nous crée deux ensembles disjoints, un au niveau de la référence éloignée, et un plus gros au niveau des références proches.



(a) Trois références proches (b) Une référence éloignée

FIGURE 17 – Visualisation de la moyenne géométrique

Cette représentation interactive est disponible derrière ce lien :

https://www.desmos.com/calculator/npx5q9nyka

5.4 Présentation de l'outil

Afin de pouvoir présenter l'outil à l'IMEC, il a fallu développer un petit outil qui pourrait simuler un filtre intégré à Autopsy/G'DIP. Cet outil doit être capable de prendre en entrée un ensemble d'images, un ensemble de références, et de renvoyer l'ensemble des images dans lesquelles la personne dont on donne les références apparaît. Il a été constaté qu'une grande partie du temps de calcul résidait dans la détection de visages et dans

l'extraction de features. Alors, afin d'accélérer l'outil, ce dernier crée pour chaque image une sauvegarde des features pour chaque visage, ainsi que sa position dans l'image.

Cet outil peut prendre un nombre variable d'arguments :

— (Obligatoire) Le dossier contenant les images à traiter
— (Obligatoire) Le dossier contenant la ou les références
— (Optionnel) La méthode d'agglomération (dans le cas d'une seule référence l'argument est sans conséquence)
— (Optionnel) Un seuil haut et un seuil bas (afin d'avoir des images très probables et probables)
— (Optionnel) Une commande d'affichage des images trouvées

De plus, une option de debug a été mise en place afin de mieux visualiser les résultats. Cette dernière donne pour chaque image la distance calculée par nos cinq fonctions de moyenne. De plus, elle crée de nouvelles images dans lesquelles les visages sont encadrés, en rouge si la distance est plus importante que le seuil haut, en bleu si la distance est entre les deux seuils, et enfin en vert si la distance est en dessous du seuil bas.

Afin de tester cet outil, nous avons récupéré un ensemble de photos prises lors d'un événement organisé par le laboratoire le 21/06/2024, la Journée du GREYC. Cet ensemble contient 72 images, dont certaines qui ne possèdent aucun visage, là où d'autres en comptent plus d'une vingtaine.

Étant donné que j'étais présent à cet événement et que j'avais trouvé une image sur laquelle j'apparaissais, je me suis dit qu'il était possible de me retrouver parmi l'ensemble des photos.

J'ai alors utilisé deux images de référence qui sont les suivantes :

FIGURE 18 – Références pour l'évaluation de l'outil

Pour ce qui est des autres paramètres, nous utilisons la moyenne géométrique, un seuil bas de 0.48 et un seuil haut de 0.54. L'algorithme nous donne alors 4 résultats, un en dessous du seuil bas et trois entre les deux seuils.

Voici les trois qui sont entre les deux seuils :

FIGURE 19 – Résultats entre les deux seuils (faux positifs) (Thomas)

Ici, les deux personnes qui sont détectées ne sont pas les bonnes, mais ont tout de même un visage assez similaire au mien. De plus, il est à noter que l'algorithme d'extraction de features ne "voit" que le contenu du rectangle, et donc ne peut pas se base sur ce qui est à l'extérieur, notamment les cheveux ou la barbe qui dépassent du rectangle sur la première image.

Ensuite, l'algorithme renvoie une autre image, qui est celle dans laquelle j'avais vu mon visage apparaître (à gauche, centre/haut). Assez surprenamment, mon visage est tout petit mais quand même très bien détecté par MTCNN, et les features très bien extraites par VGGFace.

FIGURE 20 – Résultats entre les deux seuils (vrai positif) (Thomas)

Enfin, il reste une dernière photo que l'algorithme situe en dessous du seuil bas, et qui selon moi est la plus impressionnante :

FIGURE 21 – Résultat en dessous du seuil bas (Thomas)

FIGURE 22 – Résultat en dessous du seuil bas (zoomé) (Thomas)

En effet, le visage détecté est bien le mien. Ici, le rectangle mesure 48 pixels de haut, sur 59 pixels de large, ce qui donne un total de 2832, dont environ la moitié qui ne montre même pas mon visage (qui d'ailleurs semble avoir une délimitation cohérente).

Pour ce qui est des distances, nous avons les résultats suivants :

Catégorie	Distance
Vrai positif	0.469
Faux positifs	0.524 0.532 0.516

TABLE 4 – Distance sur des images réelles (Thomas)

Enfin, pour réaliser des tests plus poussés, j'ai aussi pris comme référence le directeur du laboratoire (avec 3 références) qui apparaît, lui, sur plus de photos (et sur des photos un peu moins "complexes"). Voici une partie des images et les résultats obtenus. L'algorithme renvoie 8 résultats, 6 en dessous du seuil bas et 2 entre les deux seuils. Les calculs de distances donnent :

FIGURE 23 – Résultats en dessous du seuil bas (Directeur)

Catégorie	Distance
Vrais positifs	0.465 0.449 0.355 0.271 0.327 0.446
Faux positifs	0.520 0.508

TABLE 5 – Distance sur des images réelles (Directeur)

On remarque quelque chose d'assez intéressant : les résultats sont ici séparés. En effet, en prenant un seuil de 0.48, il est possible d'avoir une précision de 100 % (et donc un FAR et un FRR de 0). Les photos sont tout de même plus simples : la personne apparaît plus ou moins de face, et souvent l'un des / le sujet principal de la photo, ce qui fait que son visage couvre une plus grosse partie de l'image (et est d'ailleurs assez peu obstrué). Ces résultats sont plutôt rassurants, puisqu'ils confirme notre choix de seuil,

5.5 Difficultés rencontrées durant ce stage

En ce qui concerne les difficultés, on peut en citer trois principales.

La première est un souci de dataset. Comme mentionné précédemment, l'auteur d'une des deux bibliothèques que j'utilise ne semble pas vouloir mentionner la banque d'image avec laquelle il a entraîné son modèle. On peut lire dans le wiki de [3] que l'auteur a utilisé une banque avec des images qui proviennent d'internet, comme des célébrités ou autre. Il dit aussi un petit peu plus loin que le dataset est public, mais ne donne toujours pas son nom. Encore plus loin dans le wiki, il donne deux banques de données "au cas où cela serait utile pour vous" (traduction). Après quelques tests, on peut assez facilement affirmer qu'il n'a pas utilisé le second, Asian Face Age Dataset puisque les images ne sont pas labellisées en fonction des personnes mais en fonction de leur âge. Cela nous laisse penser que le premier n'a pas non plus été utilisé, [9]. De plus, cette supposition se retrouve consolidée par le fait que cette dernière soit utilisée pour faire une mesure de performances comme dans son README.md. Cependant, cela ne sont que des suppositions, et rien n'a été confirmé.

La deuxième difficulté que j'ai rencontré se situe au niveau des seuils, et plus généralement des paramètres à utiliser pour les algorithmes. En effet, puisque j'ai été contraint de n'utiliser que deux librairies, il était très facile de prendre des paramètres qui s'adaptent bien à ces banques d'images, et qui seraient mauvais pour d'autres. Cela se rapprocherait beaucoup de la loi de Goodhart, qui énonce que "lorsqu'une mesure devient un objectif, elle cesse d'être une bonne mesure". Ici, certains paramètres (et surtout le seuil) ont été choisis par rapport à ce qui fonctionnait le mieux sur la banque d'images LFW [9], et qui ne semblaient pas trop mauvais sur la deuxième banque [20]. Cependant, comme on peut le voir avec face_recognition, ce choix de paramètres aurait pu s'avérer très mauvais sur un autre jeu de données, comme sur la Journée du GREYC par exemple.

La troisième difficulté que j'ai pu rencontrer a été la plus difficile à surpasser. Lors de l'installation de VGGFace, il était écrit sur le github [17] les versions des librairies à utiliser. En l'occurrence, l'auteur suggère d'utiliser la bibliothèque keras dans sa version 2.2.4 et la bibliothèque tensorflow dans sa version 1.14.0. Au début, il m'a été impossible de télécharger tensorflow dans la bonne version, puisque celle-ci nécessitait python3.7. En utilisant un gestionnaire d'environnement, et en créant un environnement avec la bonne version de python, le problème aurait dû se résoudre. Malheureusement, même avec un nouvel environnement, la version 1.14.0 de tensorflow n'arrivait pas à se télécharger. De plus, même avec des versions proches, telles que la 1.15.0 ou encore la 1.13.0. Malheureusement, aucune ne voulait fonctionner et j'ai dû faire beaucoup d'essais sur beaucoup de versions différentes avant d'arriver à trouver une solution. D'ailleurs, en voulant récupérer les messages d'erreurs pour écrire ce rapport, j'ai réussi à installer la version 1.14.0, mais celle-ci produit encore des erreurs.

```
Traceback (most recent call last):
  File "presentation.py", line 93, in <module>
    import tensorflow as tf
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/__init__.py",
    line 28, in <module>
    from tensorflow.python import pywrap_tensorflow  # pylint: disable=unused-import
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/python/
    __init__.py", line 52, in <module>
    from tensorflow.core.framework.graph_pb2 import *
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/core/framework
    /graph_pb2.py", line 16, in <module>
    from tensorflow.core.framework import node_def_pb2 as
    tensorflow_dot_core_dot_framework_dot_node__def__pb2
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/core/framework
    /node_def_pb2.py", line 16, in <module>
    from tensorflow.core.framework import attr_value_pb2 as
    tensorflow_dot_core_dot_framework_dot_attr__value__pb2
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/core/framework
    /attr_value_pb2.py", line 16, in <module>
    from tensorflow.core.framework import tensor_pb2 as
    tensorflow_dot_core_dot_framework_dot_tensor__pb2
```

```
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/core/framework
    /tensor_pb2.py", line 16, in <module>
    from tensorflow.core.framework import resource_handle_pb2 as
    tensorflow_dot_core_dot_framework_dot_resource__handle__pb2
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/core/framework
    /resource_handle_pb2.py", line 42, in <module>
    serialized_options=None, file=DESCRIPTOR),
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/google/protobuf/
    descriptor.py", line 561, in __new__
    _message.Message._CheckCalledFromGeneratedFile()
TypeError: Descriptors cannot not be created directly.
If this call came from a _pb2.py file, your generated code is out of date and must be
    regenerated with protoc >= 3.19.0
```

En suivant les conseils données en fin de message d'erreur, on récupère une version antérieure de la bibliothèque protobuf, ce qui ne résout pas le problème.

```
Traceback (most recent call last):
  File "presentation.py", line 95, in <module>
    tf.keras.utils.disable_interactive_logging()
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/tensorflow/python/util/
    deprecation_wrapper.py", line 106, in __getattr__
    attr = getattr(self._dw_wrapped_module, name)
AttributeError: module 'tensorflow.python.keras.api._v1.keras.utils' has no attribute '
    disable_interactive_logging'
```

Afin de résoudre ce problème, j'ai décidé de prendre la dernière version disponible de tensorflow pour python3.7, la version 2.11.0. Cette version comporte quelques défauts, notamment au niveau des imports, puisqu'elle importe une version de keras qui ne correspond pas à celle utilisée par VGGFace. Nous avons cette fois cette erreur :

```
Traceback (most recent call last):
  File "presentation.py", line 101, in <module>
    from keras_vggface.vggface import VGGFace
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/keras_vggface/__init__.py
    ", line 1, in <module>
    from keras_vggface.vggface import VGGFace
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/keras_vggface/vggface.py
    ", line 9, in <module>
    from keras_vggface.models import RESNET50, VGG16, SENET50
  File "/home/zblxst/.miniconda3/envs/vgg/lib/python3.7/site-packages/keras_vggface/models.py",
     line 20, in <module>
    from keras.engine.topology import get_source_inputs
ModuleNotFoundError: No module named 'keras.engine.topology'
```

En inspectant plus profondément l'erreur, on découvre que le fichier fichier **models.py** de keras_vggface importe la fonction **get_source_inputs** du module **keras.engine.topology**, qui n'existe plus. Cependant, cette fonction existe toujours et est maintenant présente dans le module **keras.utils.layer_utils**. Une fois cette correction faite, la bibliothèque fonctionne correctement.

Enfin, étant donné qu'on m'a demandé de créer une configuration automatique, il a fallu créer un script qui télécharge et corrige à la volée le code de **keras**. Ceci peut être fait via la ligne

```
sed -i '20s/.*/from keras.utils.layer_utils import get_source_inputs/' "$(pip show
    keras_vggface | sed '8p;d' | cut -d ' ' -f 2)/keras_vggface/models.py"
```

6 Conclusion

Au cours de ce stage, nous avons comparé deux modèles de reconnaissance faciale : face_recognition et VGGFace avec MTCNN. Les résultats montrent que face_recognition offre des performances assez limitées, notamment lorsqu'il s'agit de traiter des images avec des variations d'angle, de lumière ou d'obstruction des visages. En revanche, VGGFace avec MTCNN s'en sort beaucoup mieux, offrant des résultats plus précis et fiables, même dans des conditions complexes.

L'outil de filtrage d'images que j'ai développé utilise alors VGGFace avec MTCNN, justement parce qu'il est plus performant. Cet outil permet de trier efficacement des images en fonction de la présence de visages spécifiques, ce qui est très utile pour les projets nécessitant une reconnaissance faciale fiable.

En résumé, face_recognition est moins performant dans ce cadre, mais VGGFace avec MTCNN se révèle être une solution adaptée lorsqu'on a besoin de précision pour identifier des visages dans des images. Cet outil apportera, je l'espère, une vraie aide dans le traitement semi-automatisé de grandes quantités d'images, telles que celles du projet IANEC.

Si il fallait poursuivre ce stage, une des pistes pourrait être de trouver un algorithme de classification qui permettrait de regrouper les visages des personnes, sans pour autant donner de référence à l'algorithme. Nous nous retrouverions avec une sorte de trombinoscope, qui permettrait de sélectionner un visage, et d'obtenir toutes les photos qui le contiennent.

Table des figures

Liste des tableaux

Références

[1] Souhila Guerfi Ababsa. Authentification d'individus par reconnaissance de caractéristiques biométriques liées aux visages 2d/3d. *Evry-Val d'Essonne*, 2008.

[2] Alejandro Acien, Aythami Morales, Ruben Vera-Rodriguez, Ivan Bartolome, and Julian Fierrez. Measuring the gender and ethnicity bias in deep models for face recognition. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications : 23rd Iberoamerican Congress, CIARP 2018, Madrid, Spain, November 19-22, 2018, Proceedings 23*, pages 584–593. Springer, 2019.

[3] ageitgey. face_recognition. https ://github.com/ageitgey/face_recognition. Accédé le : 2024-08-21.

[4] Jason Brownlee. How to perform face recognition with vggface2 in keras. https ://machinelearningmastery.com/how-to-perform-face-recognition-with-vggface2-convolutional-neural-network-in-keras/. Accédé le : 2024-08-21.

[5] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2 : A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.

[6] Andrea Ciamarra, Roberto Caldelli, and Alberto Del Bimbo. Temporal surface frame anomalies for deepfake video detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3837–3844, 2024.

[7] John Emert and Roger Nelson. Volume and surface area for polyhedra and polytopes. *Mathematics Magazine*, 70(5) :365–371, 1997.

[8] Ouest France. Grâce à un deepfake, les faux patrons arnaquent une entreprise de 25 millions d'euros. https ://www.ouest-france.fr/high-tech/grace-a-un-deepfake-les-faux-patrons-arnaquent-une-entreprise-de-25-millions-deuros-31e2dfb8-c38b-11ee-af40-0572f37cda9b. Accédé le : 2024-08-21.

[9] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild : A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[10] ipazc. mtcnn. https ://github.com/ipazc/mtcnn. Accédé le : 2024-08-21.

[11] iperov. Deepfacelab. Accédé le : 2024-08-21.

[12] DeepFake ENG ITA. Deepfacelab deepfake tutorial, using generic xseg. https ://www.youtube.com/watch ?v=kOIMXt8KK8M. Accédé le : 2024-08-21.

[13] Tackhyun Jung, Sangwon Kim, and Keecheon Kim. Deepvision : Deepfakes detection using human eye blinking pattern. *IEEE Access*, 8 :83144–83154, 2020.

[14] Hieu V Nguyen and Li Bai. Cosine similarity metric learning for face verification. In *Asian conference on computer vision*, pages 709–720. Springer, 2010.

[15] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[16] PimEyes. Pimeyes. https ://pimeyes.com. Accédé le : 2024-08-21.

[17] rcmalli. keras-vggface. https ://github.com/rcmalli/keras-vggface. Accédé le : 2024-08-21.

[18] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++ : Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1–11, 2019.

[19] TinEye. Tineye. https ://tineye.com/. Accédé le : 2024-08-21.

[20] trainingdata. Portrait 26 photos dataset. https ://drive.google.com/drive/folders/1nRcZurRo8aqrfITmpzBz-Y04f9BIwzHQ. Accédé le : 2024-08-21.

[21] Jonas Wagner. Forensically. https ://29a.ch/photo-forensics/. Accédé le : 2024-08-21.

[22] Wikipédia. Moyenne. https ://fr.wikipedia.org/wiki/Moyenne#Propri Accédé le : 2024-08-21.

[23] Yandex. Yandex. https ://yandex.ru/. Accédé le : 2024-08-21.

[24] Lior Yasur, Guy Frankovits, Fred M Grabovski, and Yisroel Mirsky. Deepfake captcha : a method for preventing fake calls. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, pages 608–622, 2023.

[25] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*, 23(10) :1499–1503, 2016.

7 Annexes

Project Gendarmerie
AI Spoofing in Videoconferencing

Martin Dukek, Daniel Dauksevic, Thobekile Matimbe, Nadia Lorraine Niyonsaba, Aurélien Pauger, Fatine Sennoussi, Thomas Varin

05.07.2024

Introduction

Deepfake technology has rapidly advanced, leveraging deep learning to create highly realistic audiovisual content that is challenging to distinguish from genuine material [32]. This capability raises significant concerns and potential impacts across various domains, from entertainment and advertising to misinformation and privacy violations. The development of deepfake technology builds on earlier digital manipulation techniques, such as photo and video splicing, which involved manually altering content. Unlike these traditional methods, deepfakes utilize sophisticated algorithms, making them far more convincing and harder to detect.

The motivation to understand and analyze deepfake technology stems from its dual-edged nature. On one hand, it offers innovative possibilities for creative industries, allowing for unprecedented levels of realism in visual effects and digital content creation. On the other hand, the potential for abuse in spreading false information, creating malicious content, and violating privacy is significant. These concerns necessitate a thorough exploration of the techniques used to generate deepfakes, the technological frameworks underpinning them, and the methods developed to detect and mitigate their impact.

This document explores the central aspects of video deepfake generation, beginning with an overview of the techniques used in creating deepfakes, followed by a detailed examination of the technological frameworks that support these advancements. The implementation and execution of video deepfakes are analyzed, highlighting the technical processes involved. Furthermore, we delve into the methods employed to detect video deepfakes, emphasizing both behavioral detection techniques and AI-based approaches.

Video deepfake generation

Deepfake technology leverages deep learning to generate highly realistic audiovisual content that is difficult to differentiate from genuine material [27, 31]. This section outlines the methods used to create deepfakes, building upon earlier manipulation techniques like photo/video splicing, also known as copy-move forgery. This earlier method involved manually altering images or video frames by moving or replacing sections of content [37].

Techniques

In this section, we explore deepfake generation techniques including entire face synthesis, identity swap, attribute manipulation, and expression swap. We will highlight their applications and discuss the potential implications of their use in various contexts. This overview aims to provide a clear understanding of how these technologies are applied.

Entire Face Synthesis

The entire face synthesis involves generating entirely new faces from scratch using artificial intelligence, rather than replicating or altering existing ones. To implement entire face synthesis, traditionally Generative Adversarial Networks (GANs) are the primary technology used here [35]. The generator creates images attempting to pass as real, while the discriminator evaluates their authenticity. Currently there is a sharp shift towards the more powerful Diffusion models [34].

In practice, this technique is currently most prevalent in generating digital models for advertising and creating non-existent influencers on social media platforms. That implies, the creation of entirely fictitious yet realistic-looking characters could lead to challenges in distinguishing real from synthetic in digital media, raising concerns about trust and authenticity.

Identity Swap

Commonly known as face swapping, this technique replaces one person's face in a video with another's. Autoencoders or GANs are at the core of most face swapping software [35]. In live applications such as video calls or broadcasts, the autoencoder encodes the driver's face into a latent space representation and then decodes it to match the target face. This allows for real-time face replacement, seamlessly transforming the source face into the target face [30]. This technique is especially relevant for live

applications, such as video calls or broadcasts, where real-time face replacement can occur.

In the film industry, this allows filmmakers to use the likeness of an actor without their physical presence on set. It's also used in advertising, where a brand might swap a celebrity's face onto a model's body in promotional content. The misuse of this technology in creating misleading or harmful content, such as fake news or non-consensual pornography, presents significant ethical and legal challenges.

Attribute Manipulation

The Attribute Manipulation method involves altering specific features of a face, such as age, gender, or hair color; without changing the person's overall identity. Traditionally this technique was primarily executed using GANs trained on data labeled with various facial attributes [35]. By tweaking the GAN's input parameters, desired changes are applied to the face. Though mostly used in pre-recorded content, it can subtly modify facial features in real-time to enhance realism or deceive viewers. Today, diffusion models are considered state-of-the-art for this technique [25].

Popular in social media platforms like Instagram, filters that alter facial attributes provide users with instant changes to their appearance, such as different hairstyles, makeup, or even fantastical elements like animal features. This technology is also used in fashion and beauty industries to show potential cosmetic changes. There's a risk of perpetuating unrealistic beauty standards and personal image issues, as well as the potential for deepfakes to be used in bullying or harassment.

Expression Swap

Expression swap transfers facial expressions from one video to another, superimposing them onto a different person's face. Traditionally, this was primarily achieved with a specialized type of GAN known as a conditional GAN [35]. The network is conditioned on specific facial expressions to apply them to a target face while maintaining its identity. Crucial for live applications, this technology allows for the mimicry of real-time reactions in interactive media. Currently, the focus has almost entirely shifted to diffusion models, which are now considered state-of-the-art for this technique.

Useful in the entertainment industry to enhance or alter an actor's facial expressions post-production. It's also valuable in training scenarios, such as medical simulations where realistic human reactions are necessary. Manipulating expressions can misrepresent an individual's emotional reactions, which can be misleading or damaging in interpersonal or public communications.

Regional Focus in Deepfake Manipulation

In deepfake technology, understanding the regional focus of manipulations is essential for grasping the extent of what is currently achievable and what remains challenging. This categorization sheds light on the common practices and the rarity of manipulations in different facial and upper body areas.

- **Inner Face Manipulation:** The inner face, encompassing critical features such as the eyes, nose, mouth, and eyebrows, is the most frequently targeted area in deepfake applications [12]. This focus is due to the high impact these features have on perceived identity and emotional expressions, making them central to the effectiveness of deepfakes in scenarios ranging from entertainment to misinformation.
- **Outer Face Manipulation:** Manipulation of the outer face—including the jawline, cheeks, forehead, and ears—is less common than inner face adjustments [12]. These areas are typically modified to ensure a natural integration of more central changes, enhancing the overall realism of the deepfake. However, changes here are generally subtler and complement the more detailed work done on the inner face.
- **Entire Upper Body Area Manipulation:** Manipulations that extend to the entire upper area of the body, including hair, neck, and shoulders, are quite unusual and typically reserved for high-budget productions or advanced research projects. The complexity of these manipulations stems from the need to maintain consistent movement and texture across a larger area, requiring significant computational resources and advanced modeling techniques.

It is important to understand that while deepfake technology often focuses predominantly on the inner face due to its critical role in identity and expression, outer face and entire upper area manipulations, although rarer, play a significant role in achieving full-body realism and continuity in more comprehensive applications.

Technological Frameworks in Deepfake Creation

The development and application of advanced generative models play a pivotal role in the generation of video/image deepfakes. This section delves into the two predominant types of models - GANs and Diffusion Models—examining their mechanisms, advantages, and limitations. By understanding the fundamental differences and practical implications of these technologies, we can better appreciate their impact on the authenticity and quality of deepfakes, as well as their broader implications in digital media manipulation.

GANs in Deepfakes

GANs are a class of machine learning frameworks where two neural networks contest with each other in a game [14]. In the context of deepfake generation, these two networks are:

- **Generator:** This network learns to generate plausible data. The generator's goal is to produce artificial outputs (such as images of faces) that are indistinguishable from real examples.
- **Discriminator:** This network learns to distinguish genuine data from synthetic data produced by the generator. It evaluates the authenticity of the data received, classifying it as either real or fake.

The training process for GANs involves alternating between training these two networks. The discriminator is trained to detect features that distinguish real data from data created by the generator, while the generator is trained to improve its data production to better fool the discriminator.

In deepfake technology, GANs are used to synthesize realistic human images or to alter videos by modifying specific features like facial expressions or lip movements to match an audio track. The capability of GANs to generate detailed, high-quality images makes them especially effective for creating convincing deepfakes.

Strengths

- **Realism:** GANs have been pivotal in advancing the realism of deepfakes. Their ability to generate high-quality, photorealistic images makes them particularly effective for creating convincing face swaps or identity manipulations in videos.
- **Adaptability:** GANs can be tailored for different aspects of deepfake creation, such as adjusting facial expressions or synthesizing lip movements synchronized with audio.

Weaknesses

- **Training Instability**: One of the biggest challenges with GANs in deepfake creation is their training instability [41]. This can lead to inconsistent quality in deepfakes, where some generated frames might be less realistic or show noticeable defects.
- **Artifacts and Anomalies**: GANs can sometimes introduce visual artifacts, especially under complex conditions like varying lighting, expressions, or angles. These artifacts can undermine the authenticity of deepfakes.

Autoencoders in Deepfakes

Autoencoders are a specialized type of neural network architecture used primarily for unsupervised learning of efficient codings, often applied to dimensionality reduction and feature learning [24]. In the realm of deepfakes, autoencoders play a crucial role by effectively encoding and decoding facial features or other significant elements for media manipulation. The architecture of an autoencoder can be broken down into two main components:

- **Encoder:** This part of the autoencoder compresses the input data, such as images of faces, into a more compact representation in a lower-dimensional latent space. The encoder learns to capture the essential attributes of the input data while reducing its dimensionality, which is crucial for processing efficiency and focusing on relevant features.
- **Decoder:** The decoder's role is to reconstruct the input data from its encoded form as accurately as possible. This involves translating the compressed data back into a form that closely resembles the original input, minimizing the loss of information in the process.

The training of autoencoders is driven by the goal of minimizing the reconstruction error, which enhances the network's ability to preserve essential details while ignoring noise and irrelevant variations [18]. This is particularly important in deepfake creation, where maintaining the integrity and realism of facial features is paramount.

Applications in Deepfakes Autoencoders are utilized to swap faces in videos or images by training on specific facial features and then applying this learning to modify another person's face, enabling realistic video manipulation.

Strengths of Autoencoders

- **Data Compression:** Excelling in compressing data into an essential representation, which is vital for managing large datasets in high-quality deepfakes.
- **Feature Learning:** They are adept at learning complex data patterns, essential for realistic facial transformations in deepfakes.

Weaknesses of Autoencoders

- **Blurriness in Outputs:** The reconstructed images can sometimes be blurry, lacking the sharpness of original inputs.

- **Overfitting:** Without proper regularization, autoencoders may overfit to the specific training data, reducing their effectiveness on new, unseen datasets.

Diffusion Models in Deepfakes

Diffusion models are a newer class of generative models that work differently from GANs. These models simulate a process where data starts as a random noise and gradually converts into a structured pattern through a series of steps that reverse a diffusion process [19].

- **Forward Process:** The model starts with an image of a real face and gradually adds noise to it, step by step, until the image is reduced to pure Gaussian noise.
- **Reverse Process:** The model learns to reverse this noise addition process. Starting from noise, it gradually reconstructs the original image by predicting and removing the noise at each step.

The key advantage of diffusion models over GANs is their ability to generate more detailed and high-quality images with fewer artifacts, which is crucial for the realism required in deepfakes [11]. They are also typically easier to train, as they do not require balancing two opposing networks (generator and discriminator), and they are less prone to the mode collapse problem often encountered in GAN training.

Strengths

- **Higher Fidelity:** Diffusion models generally produce images and videos with greater detail and fewer noticeable artifacts compared to GANs. This is crucial in maintaining the illusion in deepfakes, particularly for high-definition content [11].
- **Robustness to Overfitting:** Diffusion models, through their iterative refinement process, are less prone to overfitting compared to GANs. This can lead to more consistent and reliable results in diverse conditions.

- **Computational Intensity:** The iterative nature of diffusion models, requiring many steps to generate an image, makes them computationally intensive. This is a significant drawback for real-time or near-real-time applications such as live streaming deepfakes.
- **Latency in Generation:** The slow generation process can be a limiting factor, making diffusion models less suitable for applications where quick turnaround is crucial, like interactive media or live modifications in video calls.

Comparison and Usage in Deepfake Generation

While both GANs and diffusion models are powerful tools for image generation, diffusion models are gaining popularity in the field of deepfake creation due to their superior image quality and training stability. However, GANs still remain popular due to their relative efficiency and established track record in various applications.

In practice, the choice between using a GAN or a diffusion model often depends on the specific requirements of the project, such as the desired quality of the output, the computational resources available, and the specific type of manipulation (e.g., face swapping, expression editing) needed.

By integrating either of these technologies, deepfake creators can achieve varying levels of realism and fidelity, which can be used for legitimate purposes like filmmaking, advertising, and virtual reality, or for malicious purposes like creating misleading media and impersonation.

Implementation of Video Deepfakes

This section examines the technical steps involved in creating video deepfakes, focusing primarily on the techniques of 'identity swap' and 'attribute manipulation.' While identity swaps are particularly prevalent in live video applications due to their dynamic nature, attribute manipulation plays a crucial role in both live and pre-recorded settings. We will detail the step-by-step processes essential for executing these techniques effectively, emphasizing the technological steps required to maintain realism and address the potential challenges and implications associated with their use.

Technical Execution of Identity Swaps in Live Video

The process of implementing identity swaps in live video involves several intricate steps, each important for achieving a convincing swap.

1. **Facial Detection and Tracking:** The first step in a live identity swap is detecting and continuously tracking the facial features of both the source (whose features are being copied) and the target (who will receive the features). This is

done using real-time facial detection algorithms, e.g. FAN [39], that can identify and locate faces within video frames despite movement or changes in expression.

2. **Facial Landmark Localization:** Once faces are detected, the next step involves pinpointing specific facial landmarks on both the source and target faces. These landmarks typically include the eyes, nose, mouth, and jawline. Advanced techniques involve up to 68 landmarks or more for greater precision. Real-time accuracy in landmark localization is crucial, as it directly impacts the quality of the face swap.
3. **Feature Extraction and Encoding:** The identified landmarks are used to extract facial features, which are then encoded into a data format that can be manipulated. In live scenarios, streamlined autoencoders are commonly used for this purpose. These autoencoders compress the facial data into a lower-dimensional space, capturing the essential characteristics of each face while minimizing data size for quicker processing.
4. **Real-Time Processing and Swapping:** The encoded features of the source face are swapped onto the target face in real time. This involves adjusting the encoded data to align with the facial structure and orientation of the target. The swap needs to account for differences in facial geometry, expression, and orientation to avoid misalignments and preserve the natural dynamics of the face.
5. **Rendering and Blending:** After the swap, the modified facial features are rendered back onto the video stream of the target. This step must be seamless, with careful blending to match the skin tone, lighting, and textural details of the target. The blending also needs to be dynamic, adjusting continuously as the video feeds change due to movement or environmental factors.
6. **Output and Adjustment:** The final output is a live video where the target's face incorporates the source's facial features. This output needs to be continuously monitored and adjusted to ensure consistency and realism. Adjustments might be needed to account for latency, synchronization issues, or quality degradation during the transmission.

Technical Execution of Expression Swaps in Prerecorded Videos

Manipulating prerecorded videos often involves expression swaps, which benefit from the controlled environment of non-live footage, reducing overlay errors and minimizing the discrepancies caused by rapid motion. This technique allows for more precise

alignment and integration of new expressions, enhancing realism and consistency throughout the video.

1. **Video Analysis and Frame Selection:** The first step is a thorough analysis of the video to select key frames where facial expressions will be modified. Frames are chosen based on the clarity and stability of the face, ensuring consistent lighting and minimal motion blur for optimal results.
2. **Facial Recognition and Feature Tracking:** Advanced facial recognition technology is employed to track facial features throughout the selected frames. This tracking is crucial to maintain accurate alignment of facial features as expressions change, ensuring continuity and fluidity in facial movements across the video.
3. **Expression Modeling and Simulation:** Using deep learning models, the target expressions are generated based on a predefined set of emotions or custom inputs. These models can simulate realistic facial expressions by adjusting key features such as the eyebrows, eyes, and mouth in a way that corresponds with the original face's musculature and emotional context.
4. **Expression Integration and Blending:** The new expressions are then integrated into the video frames using sophisticated blending techniques. This involves careful manipulation of facial textures and dynamics to ensure that the new expressions blend seamlessly with the original footage, maintaining natural lighting, shadow, and texture continuity.
5. **Rendering and Quality Assurance:** After the expressions are integrated, the video is rendered to incorporate the changes smoothly. This step often includes a quality assurance process where the video is reviewed frame-by-frame to ensure that the expression swaps do not introduce any visual anomalies or disrupt the video's overall flow.
6. **Output and Review:** The final output is a version of the original video with altered expressions. This video is typically reviewed in multiple scenarios and on various devices to ensure the changes are effective and realistic under different viewing conditions.

Detecting Video Deepfakes

Detecting deepfakes is crucial in mitigating the spread of misleading information. The effectiveness of detection correlates with the time and resources available to analyze the content before it influences behaviors or decisions. This is especially pertinent in deepfake video conferencing, where the sophistication of the fake directly impacts both its believability and detectability.

Behavioural Detection Techniques

Detecting deepfakes in video calls for close observation, leveraging both technical cues and behavioral inconsistencies. Live video streams, subject to real-time processing limitations, might display noticeable computational lags or artifacts—blurry edges or delayed responses—that indicate signal tampering. In contrast, prerecorded videos allow creators to polish these imperfections, necessitating a more nuanced detection approach focusing on subtleties like frame-by-frame discrepancies or audio-visual mismatches.

Detection in Live Videos Deepfake technology, when applied to live video streams, can encounter unique challenges due to the real-time processing requirements. Detecting deepfakes in this context can often be facilitated by observing specific behavioral and technical cues:

Computational Delays and Artifacts: Given the computationally intensive nature of generating deepfakes live, delays in video stream or artifacts such as blurring at the edges of the superimposed face are common. These artifacts typically appear where the fake face overlays the real background.

Inconsistent Facial Tracking: Rapid movements or changes in facial expression can disrupt the deepfake's facial mapping, causing anomalies. Quick head movements, like turning the head to a 90-degree angle, often reveal misalignments that are not consistent with natural movements [38].

Interaction with Objects: Actions such as waving a hand in front of the face may interfere with the deepfake's ability to maintain consistent facial tracking and rendering, making the artificial nature of the video apparent [38].

Detection in Prerecorded Videos Prerecorded deepfake videos offer more control over the final output, reducing some of the errors seen in live deepfakes. However, they still exhibit specific weaknesses that can be identified through careful observation:

Frame-by-Frame Analysis: Detailed examination of video frames can uncover inconsistencies in lighting, texture, or edge artifacts where the deepfake has been applied. Slow motion playback can enhance the visibility of these discrepancies.

Facial Landmark Consistency: Inconsistencies in facial landmarks across consecutive frames can indicate manipulation. This includes abnormal blinking patterns or asymmetric facial movements that do not align with typical human expressions.

Audio-Visual Mismatch: Often, the synchronization between audio and visual elements can be off in deepfakes. Discrepancies in lip-syncing or facial expressions that do not appropriately match spoken words are common indicators.

AI-Based Detection Techniques

While neural networks represent the most promising avenue for detecting deepfakes, they currently face significant challenges in generalization. These detection models tend to perform well on the specific datasets they were trained on but struggle to maintain accuracy across diverse or unseen data. This limitation stems partly from the variability in human expressions and behaviors, such as what constitutes an irregular blinking pattern, which can differ widely among individuals and contexts. This specificity issue highlights a critical gap in current AI capabilities, which is adapting to the broad range of deepfake generation techniques continuously evolving in complexity and realism.

1. **Facial Expressions and Head Movements:** This method leverages neural networks to analyze the fluidity and accuracy of facial expressions and head movements [1, 6]. The technique focuses on identifying subtle discrepancies that do not match typical human motions, relying on a database of known patterns to spot anomalies. However, variations due to cultural or individual uniqueness can complicate the detection process, as these might be misinterpreted as manipulations by the system.

2. **Blinking Patterns Analysis:** This approach examines the frequency and naturalness of blinking, a subtle human behavior that deepfake algorithms often neglect [21, 26]. By analyzing how and when a person blinks, AI models can pinpoint inconsistencies typical of synthetic manipulation. However, as deepfake technology advances, creators are beginning to model more realistic blinking, thus requiring continuous updates to the detection algorithms to maintain their

effectiveness.

3. **Artifact/Anomaly Detection:** This method scans for technical flaws such as pixel anomalies, color mismatches, and improper edge integrations, which are common byproducts of the deepfake generation process [4, 5]. High-quality deepfakes are becoming adept at minimizing these imperfections, which necessitates a more nuanced approach to detect ever-smaller inconsistencies that may indicate tampering.

4. **Mesoscopic and Steganalysis Features:** Employing a detailed examination of image textures and embedded data, this method digs into mesoscopic features and steganalysis to reveal alterations [28, 17]. The analysis includes checking for consistency in texture and looking for hidden data anomalies that might emerge from the manipulation process. This technique demands high computational power and faces challenges in keeping up with high-resolution deepfakes that increasingly obscure such micro-signatures.

The ongoing struggle to effectively detect deepfakes is significantly complicated by the neural networks' inability to generalize across varying datasets and the rapid evolution of deepfake technologies. This rapid pace of advancement means that once detection methods are developed and possibly shared as open-source software, they quickly become outdated as new deepfake generation methods are devised that circumvent earlier detection algorithms. As a result, there's a notable absence of reliable, scalable open-source solutions for deepfake detection, as the field demands continuous, dynamic updating of detection technologies to keep pace with the sophistication of deepfake creators.

Here's how you can update and conclude the final subsection by incorporating the rise of diffusion models and the specific challenges of live face transplants focused on the inner face:

Feasibility of Deepfakes: Live vs. Prerecorded

Deepfake technology, particularly in the context of its application in live versus prerecorded settings, presents distinct challenges and risks. With the rise of advanced generative models such as diffusion models, the landscape of deepfake technology is evolving rapidly. Each method—live and prerecorded—has its own technical requirements, feasibility issues, and potential dangers.

Emergence of Diffusion Models

The introduction of diffusion models has marked a significant advancement in the field of deepfakes. These models, known for generating high-quality images through a process of adding and removing noise, offer greater detail and realism compared to traditional methods like GANs. Their ability to refine and enhance the quality of deepfakes makes them particularly influential in the realm of prerecorded deepfakes, where creators have the luxury of time to perfect the output.

Live Deepfakes

Live deepfakes, which involve real-time video manipulation, are technologically demanding and still relatively nascent. Key considerations include:

- **Computational Demand:** Executing deepfakes live requires powerful computing resources to handle real-time processing, making it less accessible.
- **Quality and Stability:** Live deepfakes struggle with maintaining consistent quality, especially under variable conditions like changes in lighting or facial expressions.
- **Inner Face Manipulation:** Most live face transplants focus primarily on the inner face—eyes, nose, mouth—which requires the driver (source face) to have physical similarities to the target to avoid issues like 'face bleed,' where edges of the transplanted face mismatch with the underlying features.

Despite these limitations, the potential misuse of live deepfake technology, particularly in applications like video conferencing, poses significant risks that necessitate continuous monitoring and advancement in real-time detection technologies.

Prerecorded Deepfakes

Prerecorded deepfakes are more advanced and pose a greater threat due to their refined and indistinguishable nature from authentic content. They are characterized by:

- **Advanced Editing Tools:** Access to sophisticated video editing software allows for detailed manipulation and enhancement of deepfakes.
- **Time for Refinement:** Creators can spend significant time perfecting the appearance and plausibility of prerecorded deepfakes, eliminating most detectable flaws.

- **Skillset Requirement:** Producing high-quality prerecorded deepfakes requires advanced skills in video editing and machine learning, making it a specialized yet potentially malicious tool in skilled hands.

Prerecorded deepfakes are especially dangerous due to their potential use in creating fake news or impersonating individuals in damaging ways. This capability underscores the urgent need for effective detection and regulation strategies.

Conclusion

The feasibility and dangers of deepfakes vary significantly between live and prerecorded forms. While the technology for live deepfakes continues to develop amidst significant challenges, prerecorded deepfakes currently represent a more immediate and profound threat due to their sophistication and potential for misuse. As diffusion models and other advanced technologies continue to evolve, the landscape of deepfakes will most likely grow more complex.

Tooling

In this section, we introduce three widely-used tools for the creation of deepfakes. These tools leverage advanced computer vision and deep learning techniques to perform real-time face-swapping and editing in video streams and static images. Each tool has contributed significantly to the field, providing capabilities ranging from live video manipulation to detailed pre-recorded face swaps, thus enabling a variety of applications in entertainment, media, and more.

DeepFaceLive

DeepFaceLive (https://github.com/iperov/DeepFaceLive) is mature opensource real-time face-swapping application that has been active since 2021. The software leverages deep learning and GANs to perform realistic and seamless face replacements in live video streams. This tool was a significant advancement in the field of computer vision and synthetic media, enabling applications ranging from entertainment to social media content creation.

Functionality

At its core, DeepFaceLive performs the complex task of replacing one face with another in real-time video streams. The process is composed of several key stages, each involving advanced techniques from computer vision and deep learning:

1. **Face Detection and Tracking:**

   Utilizes the Face Alignment Network (FAN), a specialized deep learning model designed to detect and precisely locate key facial landmarks in real-time [3]. FAN uses 68 unique facial landmarks to achieve its task.

2. **Feature Extraction and Alignment:** Key facial landmarks are identified on both the source (face to be swapped) and target (face in the video) faces. These landmarks typically include the eyes, nose, mouth, and facial contours. The source face is then geometrically transformed and aligned to match the pose, orientation, and expression of the target face. This alignment ensures that the swapped face maintains natural movements and expressions.

3. **Image Generation:** Autoencoders are used to perform the face synthesis and manipulation. The encoder network compresses the facial images into a lower-dimensional latent space by learning from a large dataset of facial images. It aims to capture the most significant features necessary for face reconstruction. The decoder network then attempts to reconstruct the face from

this encoded representation, focusing on producing images that closely resemble the original faces. The training process involves minimizing the difference between the original and reconstructed images, which helps the autoencoder produce somewhat realistic face swaps.

4. **Inner Face Replacement:** DeepFaceLive focuses on inner face replacement, targeting the key facial features (eyes, nose, mouth) while preserving the outer contours and background of the original video frame. This approach ensures that the new face blends naturally into the existing video, maintaining consistency in lighting, color, and texture.
5. **Post-Processing:** Post-processing techniques are employed to refine the face swap, addressing issues such as color mismatches, lighting discrepancies, and edge artifacts. Techniques such as color correction, edge smoothing, and blending help achieve a seamless and convincing final output.

Faceswap

Faceswap (https://github.com/deepfakes/faceswap) is an open-source software tool (actively developed since 2017) for creating realistic face swaps in videos and images, actively developed since the emergence of deepfake technology. This application employs machine learning and neural network methodologies, specifically designed to manage both pre-recorded content and batch processing scenarios.

Faceswap is often recognized as the first application that brought deepfake technology to a broad public, simplifying the creation of realistic face swaps. It has gained notoriety, particularly for its misuse in creating non-consensual sexually explicit content.

Functionally, it works similar to DeepFaceLive, but the software places a much greater emphasis on producing highly convincing pre-recorded deepfakes.

Functionality [36][20]

1. **Lightweight Model** The Lightweight model is designed for users with lower-end GPUs ($\leq$ 2GB VRAM). The model's lightweight architecture is in this way designed to minimize VRAM usage. It can be used for quick and efficient training on higher end GPUs which is useful for initial testing of face swaps before using more resource-intensive models. According to the code in `lightweight.py`, it uses a basic convolutional layer (`Conv2D`) followed by a max-pooling layer (`MaxPooling2D`):

```
def build_encoder(self):
    model = Sequential()
    model.add(Conv2D(32, (5, 5), activation='relu', input_shape=(64, 64, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    return model
```

2. **Original Model**

   The Original model, as the foundational model of Faceswap, is designed to provide a balance between complexity and performance. It can produce high-quality results, especially when high-quality datasets are used. The code in `original.py` reveals that this model employs a more comprehensive set of layers compared to the Lightweight model:

```
def build_encoder(self):
    model = Sequential()
    model.add(Conv2D(64, (5, 5), activation='relu', input_shape=(64, 64, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(128, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    return model
```

   Here, the encoder uses multiple convolutional layers with increasing filter sizes (64 and 128 filters), each followed by max-pooling layers. This setup allows for more detailed feature extraction and a more complex representation of the input data.

3. **IAE (Intermediate Auto-Encoder) Model**

   The IAE model has a slightly different structure to better separate identity by including intermediate layers that sit between the encoder and decoder.

   This model introduces three intermediate layers to better separate identity features. It is structured with a shared encoder and decoder but includes three intermediate layers—one for input A, one for input B, and one shared—between the encoder and decoder. This architecture aims to enhance the distinction between different faces. The code in `dfl_sae.py` shows the presence of these intermediate layers:

```
def build_intermediate_layers(self):
    self.intermediate_A = Dense(512, activation='relu')
    self.intermediate_B = Dense(512, activation='relu')
    self.intermediate_shared = Dense(512, activation='relu')
```

4. **Dfaker Model**

   The Dfaker model focuses on upscaling from lower to higher resolution, for more enhanced output quality. It uses techniques different from the Original model and provides a straightforward approach with input sizes ranging from 64px to 128px and output sizes from 128px to 256px. The code in `dfaker.py` highlights the model's upscaling capabilities through the use of `UpSampling2D` layers:

```
def build_upsampler(self):
    model = Sequential()
    model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(64, 64, 3)))
    model.add(UpSampling2D(size=(2, 2)))
    return model
```

5. **Unbalanced Model**

   The Unbalanced model is highly customizable and powerful but requires more expertise to achieve good results. It handles input and output sizes ranging from 64px to 512px, with a greater emphasis on the B decoder, meaning reversing a swap may lead to less satisfactory results.

6. **DFL-H128 Model**

   The DFL-H128 model uses the same encoder and decoder as the Original model but processes 128px input images, compressing the image into a smaller latent space.

7. **DFL-SAE Model**

   The DFL-SAE model provides a variety of parameters and settings that users can adjust to fine-tune the deepfake creation process. These options include choices related to the neural network architecture, training parameters (such as learning rate and batch size), and specific techniques to enhance the quality of the deepfake (like resolution, layers, and iterations).

   The DFL-SAE model includes two different network structures: one based on the Original model's shared encoder/split decoders and another based on the IAE model's shared intermediate layers. The code in `dfl_sae.py` illustrates this dual structure:

```
class DFL_SAE:
    def build_models(self):
        # Shared encoder
        self.encoder = self.build_encoder()
        # Split decoders
```

```
self.decoder_A = self.build_decoder('A')
self.decoder_B = self.build_decoder('B')
```

Despite the ability to generate detailed results, there is a potential issue of "identity bleed" — that is, the phenomenon where some features or attributes of Person A (the source face) remain noticeable in the face of Person B (the target face) after the face swap. This can happen due to the inherent complexity of completely isolating and transferring only the desired facial features without inadvertently transferring some unwanted characteristics. For example, subtle expressions, skin texture, or lighting conditions specific to Person A may still be slightly visible in the face of Person B, which can reduce the effectiveness of the face swap and make the result appear less authentic.

8. **Villain Model**

   The Villain model is known for its high detail but is very VRAM intensive. It typically uses a 128px input and output size. The Villain model is highly detailed and VRAM intensive, suitable for high-resolution swaps without extensive customization options.

9. **Realface Model**

   The Realface model is a successor to the Unbalanced model, supporting input sizes from 64px to 128px and output sizes from 64px to 256px. It is highly customizable, allowing experienced users to tweak settings for optimal results

10. **Dlight Model**

    The Dlight model is a higher resolution model based on the Dfaker variant, focusing on upscaling faces with custom upscalers. It supports a 128px input and output sizes ranging from 128px to 384px. The code in `dlight.py` includes specific upscaling blocks:

```
def build_upsampler(self):
    model = Sequential()
    model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(128, 128, 3)))
    model.add(UpSampling2D(size=(2, 2)))
    return model
```

11. **Phaze-A Model** [20]

    This is the latest model for Faceswap. The Phaze-A model is a very complex model builder suitable for advanced users. It offers more flexibility and customization options. Unlike previous models, Phaze-A is designed to allow users to configure almost every aspect of the model, enabling the creation of models

tailored to specific needs and hardware capabilities. This model allows the construction of networks that can run on less than 1GB of VRAM or require substantial computational resources. This adaptability means that the model can be optimized for a wide range of hardware configurations. Phaze-A's architecture includes shared weights for the encoder and customizable sub-models for other components, such as the bottleneck and decoder.

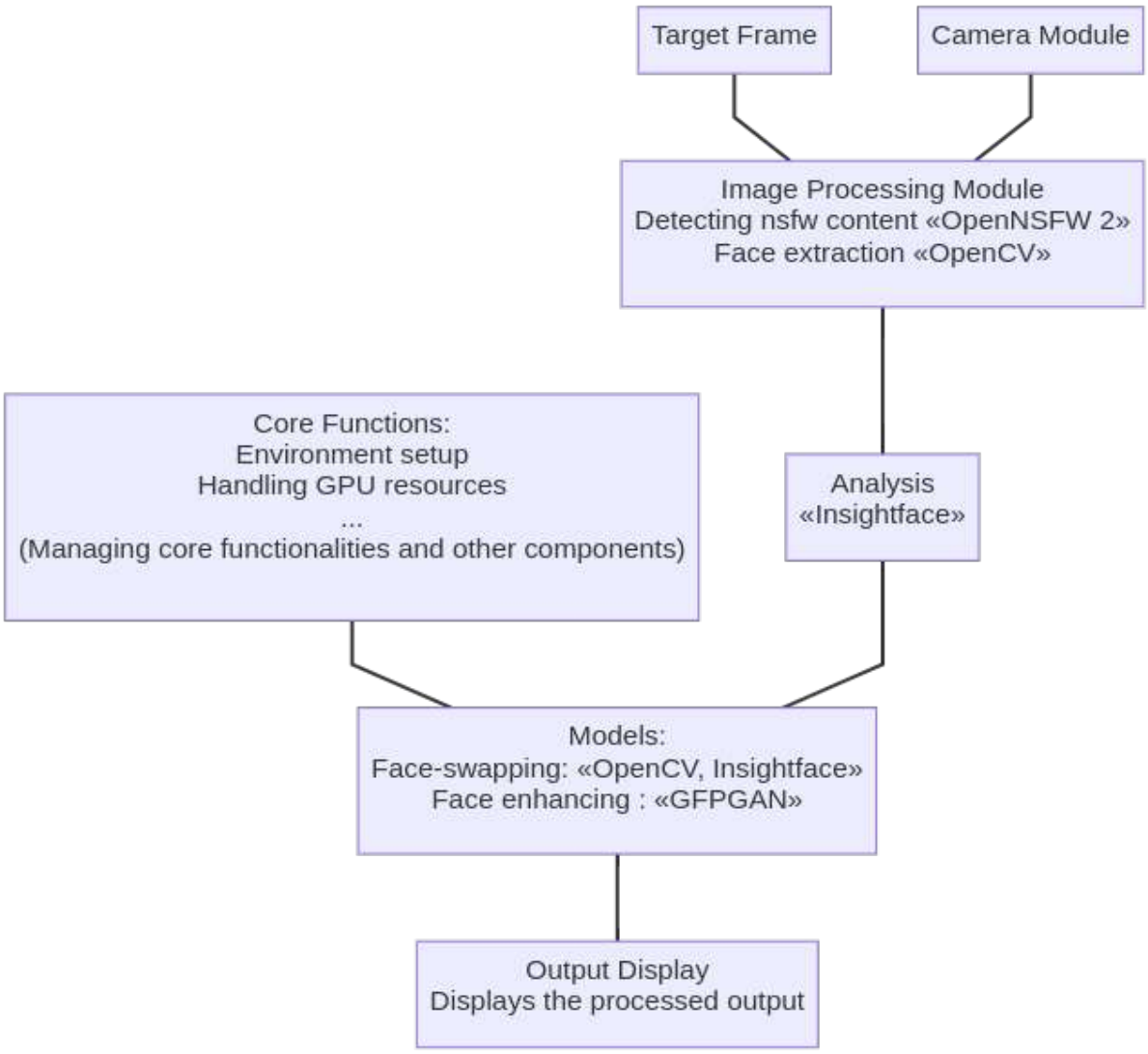
The G-Block, inspired by Nvidia's StyleGAN added enhanced lighting and fine detail reproduction. The model's documentation stresses that Phaze-A is not a "fire and forget solution"; users need to understand and tweak the settings to achieve optimal results.

Deep-Live-Cam

Deep-Live-Cam is an open-source application designed for real-time face swapping, actively developed since 2023 and accessible via its GitHub repository.

Similar to DeepFaceLive, Deep-Live-Cam carries out the task of face replacement in real-time video streams through several key processes. Each process involves methodologies from computer vision and deep learning that aims at achieving accurate and "believable" face swaps.

Functionality



1. **Preprocessing**

   The capturer module leverages the OpenCV library to interface with video files. By managing these low-level operations, it ensures that the system has access to specific video frames required for further processing.

NSFW content detection The `predict_image` function checks if a given image file contains NSFW content using the OpenNSFW model. It processes the image and returns `True` if the NSFW probability exceeds the threshold.

From the `ui.py` snippet:

The `update_preview` function includes a call to `predict_frame` using the model, but a simple check afterwards of NSFW content in the target frames during preview updates. If NSFW content is detected, the function quits the process:

```
if modules.globals.nsfw == False:
    from modules.predicter import predict_frame
    if predict_frame(temp_frame):
        quit()
```

And thus, the preview (and likely further processing) is halted to prevent inappropriate content from being processed or displayed. In general, the exact handling mechanism would depend on how the NSFW detection results are integrated into the broader application logic. However, the source code of this open-source application can be altered to bypass this check for potential inappropriate use.

2. **Analysis and Face Extraction**

   This process begins with face detection, which employs pre-trained models from the OpenCV library (Haar cascades) to identify facial regions within the frames. Once faces are detected, the module proceeds to extract these regions, isolating the faces from the rest of the frame.

   The `get_face_analyser` function initializes and configures the InsightFace-based face analysis tool if it hasn't been set up already, creating a global `FACE_ANALYSER` object with specified model settings and context.

   This function ensures that the face analyzer is ready for use by creating an instance of `insightface.app.FaceAnalysis` with the specified model and execution settings, such as GPU providers if configured. During face extraction, functions like `get_one_face` and `get_many_faces` use the face analyzer to detect faces within a frame.

3. **Deep Learning Model**

   The Deep Learning Model process in Deep-Live-Cam involves two main components: face enhancement using the GFPGAN model and face swapping using the InsightFace model.

The `enhance_face` in`face_enhancer.py` function uses the GFPGAN model to improve the resolution and quality of detected faces in a frame.

The `swap_face` function in `face_swapper.py` then uses the InsightFace model to swap a detected source face with a target face within a frame. This involves aligning the faces and replacing the target face with the source face which enables real-time face replacements in video streams.

4. **Core Functions**

   The software leverages environment configuration and dynamic allocation of resources based on the hardware capabilities detected at runtime. GPU usage in Deep-Live-Cam is configurable via command-line arguments (`--execution-provider`) in the `core.py` script. The available execution providers, such as CPU and CUDA, are determined using the ONNX Runtime. Environment variables and TensorFlow configurations are set to utilize the GPU if specified. For example, setting `OMP_NUM_THREADS` to `1` optimizes CUDA performance, and configuring TensorFlow to grow GPU memory as needed ensures efficient resource usage. Both the face enhancement and face swapping models can utilize the GPU if the execution provider is set to CUDA, for accelerated processing and improved efficiency in real-time applications.

Limitations

DeepFaceLive and Deep-Live-Cam have several limitations:

- **Similarity Requirement:** For believable results, the driver (source) face and victim (target) face need to have similar facial structures and features. Significant differences in facial shape, skin tone, or feature placement can lead to unnatural or unconvincing results. Additionally, the replacement is limited to the inner face, meaning jaw, ear and hair will not be replaced.
- **Comparatively Weak Face Tracking:** FANs comparatively low number of facial landmarks makes it weaker at tracking the driver face through obfuscation and at steeper angles than more modern face tracking software [3].
- **Real-Time Processing Constraints:** Achieving real-time performance requires substantial computational power, typically leveraging high-end GPUs. This can limit accessibility for users without powerful hardware.
- **Lighting and Environmental Conditions:** Variations in lighting and environmental conditions between the source and target faces can pose challenges. The software may struggle to seamlessly blend faces if there are stark differences

in lighting or background.

- **Expression and Movement Matching:** While GANs and alignment algorithms can accurately match facial expressions and movements, rapid or complex movements can still present challenges, potentially resulting in artifacts or lag.

References

[1] Shruti Agarwal and Hany Farid. "Detecting Deep-Fake Videos From Aural and Oral Dynamics". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2021, pp. 981–989.

[2] Xiang An et al. "Partial FC: Training 10 Million Identities on a Single Machine". In: *Arxiv 2010.05222*. 2020.

[3] Adrian Bulat and Georgios Tzimiropoulos. "How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks)". In: *International Conference on Computer Vision*. 2017.

[4] Andrea Ciamarra, Roberto Caldelli, and Alberto Del Bimbo. "Temporal Surface Frame Anomalies for Deepfake Video Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2024, pp. 3837–3844.

[5] Andrea Ciamarra et al. "Deepfake Detection by Exploiting Surface Anomalies: The SurFake Approach". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*. Jan. 2024, pp. 1024–1033.

[6] Ilke Demir and Umur Aybars Ciftci. "Where Do Deep Fakes Look? Synthetic Face Detection via Gaze Tracking". In: *ACM Symposium on Eye Tracking Research and Applications*. ETRA '21 Full Papers. Virtual Event, Germany: Association for Computing Machinery, 2021. ISBN: 9781450383448. DOI: 10.1145/3448017.3457387. URL: https://doi.org/10.1145/3448017.3457387.

[7] Jiankang Deng et al. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". In: *CVPR*. 2019.

[8] Jiankang Deng et al. "RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild". In: *CVPR*. 2020.

[9] Jiankang Deng et al. "Sub-center ArcFace: Boosting Face Recognition by Large-scale Noisy Web Faces". In: *Proceedings of the IEEE Conference on European Conference on Computer Vision*. 2020.

[10] Jiankang Deng et al. "The Menpo benchmark for multi-pose 2D and 3D facial landmark localisation and tracking". In: *IJCV* (2018).

[11] Prafulla Dhariwal and Alexander Nichol. "Diffusion Models Beat GANs on Image Synthesis". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 8780–8794. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf.

[12] Jie Gao et al. "Generalized Deepfake Detection Algorithm Based on Inconsistency Between Inner and Outer Faces". In: *Image Analysis and Processing - ICIAP*

2023 Workshops. Ed. by Gian Luca Foresti, Andrea Fusiello, and Edwin Hancock. Cham: Springer Nature Switzerland, 2024, pp. 343–355. ISBN: 978-3-031-51023-6.

[13] Baris Gecer, Jiankang Deng, and Stefanos Zafeiriou. "OSTeC: One-Shot Texture Completion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[14] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[15] Jia Guo et al. "Sample and Computation Redistribution for Efficient Face Detection". In: *arXiv preprint arXiv:2105.04714* (2021).

[16] Jia Guo et al. "Stacked Dense U-Nets with Dual Transformers for Robust Face Alignment". In: *BMVC*. 2018.

[17] Zhiqing Guo et al. "Exposing Deepfake Face Forgeries With Guided Residuals". In: *IEEE Transactions on Multimedia* 25 (2023), pp. 8458–8470. DOI: 10.1109/TMM.2023.3237169.

[18] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647. eprint: https://www.science.org/doi/pdf/10.1126/science.1127647. URL: https://www.science.org/doi/abs/10.1126/science.1127647.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[20] *Introducing - Phaze-A*. Faceswap Forum. Section: Training Discussion. Apr. 14, 2021. URL: https://forum.faceswap.dev/viewtopic.php?f=27&t=1525 (visited on 07/04/2024).

[21] Tackhyun Jung, Sangwon Kim, and Keecheon Kim. "DeepVision: Deepfakes Detection Using Human Eye Blinking Pattern". In: *IEEE Access* 8 (2020), pp. 83144–83154. DOI: 10.1109/ACCESS.2020.2988660.

[22] Jeremy Kahn. *Business execs are facing a new threat: the person talking on Zoom might be an A.I.-generated 'deep fake.' Here are some easy ways to tell if you're talking to a fake*. Sept. 2022. URL: https://fortune.com/2022/09/03/live-deepfakes-detect-methods-zoom-fraud/.

[23] Rahul Katarya and Anushka Lal. "A study on combating emerging threat of deepfake weaponization". In: *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE. 2020, pp. 485–490.

[24] Jan Kietzmann et al. "Deepfakes: Trick or treat?" In: *Business Horizons* 63.2 (2020). ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING, pp. 135–146. ISSN: 0007-6813. DOI: https://doi.org/10.1016/j.bushor.2019.11.006. URL: https://www.sciencedirect.com/science/article/pii/S0007681319301600.

[25] Chaerin Kong et al. “Leveraging Off-the-Shelf Diffusion Model for Multi-Attribute Fashion Image Manipulation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2023, pp. 848–857.

[26] Romeo Lanzino et al. “Faster Than Lies: Real-time Deepfake Detection using Binary Neural Networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2024, pp. 3771–3780.

[27] Y. Patel et al. “Deepfake generation and detection: Case study and challenges”. In: *IEEE Access* 11 (2023), pp. 143296–143323.

[28] Yogesh Patel et al. “An Improved Dense CNN Architecture for Deepfake Image Detection”. In: *IEEE Access* 11 (2023), pp. 22081–22095. DOI: 10.1109/ACCESS.2023.3251417.

[29] Yogesh Patel et al. “Deepfake generation and detection: Case study and challenges”. In: *IEEE Access* (2023).

[30] Ivan Perov et al. *DeepFaceLab: Integrated, flexible and extensible face-swapping framework*. 2021. arXiv: 2005.05535 [cs.CV]. URL: https://arxiv.org/abs/2005.05535.

[31] K. N. Ramadhani and R. Munir. “A comparative study of deepfake video detection method”. In: *Proc. 3rd Int. Conf. Inf. Commun. Technol. (ICOIACT)*. Nov. 2020, pp. 394–399.

[32] Md Shohel Rana et al. “Deepfake detection: A systematic literature review”. In: *IEEE access* 10 (2022), pp. 25494–25513.

[33] Xingyu Ren et al. “Facial Geometric Detail Recovery via Implicit Representation”. In: *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*. 2023.

[34] Michal Stypulkowski et al. “Diffused Heads: Diffusion Models Beat GANs on Talking-Face Generation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 5091–5100.

[35] Ruben Tolosana et al. “Deepfakes and beyond: A Survey of face manipulation and fake detection”. In: *Information Fusion* 64 (2020), pp. 131–148. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2020.06.014. URL: https://www.sciencedirect.com/science/article/pii/S1566253520303110.

[36] *Training in Faceswap*. Faceswap Forum. Section: Training Discussion. Sept. 30, 2019. URL: https://forum.faceswap.dev/viewtopic.php?t=146&source=post_page-----8084e6bf01e0---------- (visited on 07/04/2024).

[37] Nor Bakiah Abd Warif et al. “Copy-move forgery detection: Survey, challenges and future directions”. In: *Journal of Network and Computer Applications* 75 (2016), pp. 259–278. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.

jnca.2016.09.008. URL: https://www.sciencedirect.com/science/article/pii/S1084804516302144.

[38] Lior Yasur et al. "Deepfake CAPTCHA: A Method for Preventing Fake Calls". In: *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*. ASIA CCS '23. Melbourne, VIC, Australia: Association for Computing Machinery, 2023, pp. 608–622. ISBN: 9798400700989. DOI: 10.1145/3579856.3595801. URL: https://doi.org/10.1145/3579856.3595801.

[39] Xi Yin et al. *FAN: Feature Adaptation Network for Surveillance Face Recognition and Normalization*. 2020. arXiv: 1911.11680 [cs.CV]. URL: https://arxiv.org/abs/1911.11680.

[40] Peipeng Yu et al. "A survey on deepfake video detection". In: *Iet Biometrics* 10.6 (2021), pp. 607–624.

[41] Baiwu Zhang et al. *On Attribution of Deepfakes*. 2021. arXiv: 2008.09194 [cs.LG]. URL: https://arxiv.org/abs/2008.09194.