

Projet Thématique

Système intelligent de reconnaissance faciale en temps réel



Spécialité : 2eme année instrumentation et systèmes embarqués

Année : 2023 – 2024

Encadré par : Pr. Anissa MOKRAOUI

Membre de groupe : Iheb BOUARICHE | Arselan MEGHELLI

Table des matières

1. Introduction :	3
1.1. Définition du projet :	3
1.2. Cahier de charges :	3
1.3. Le matériel nécessaire pour la réalisation de ce projet :	3
2. Stratégie de travail :	3
2.1. Répartition des tâches :	4
2.2. Méthodologie de travail :	4
2.3. Changement de stratégie :	4
2.4. Solution validée :	5
3. Fonctionnement de la partie Software :	5
3.1. Extraction et détection des visages :	5
3.1.1. Teste pour différentes méthodes :	5
3.1.2. Notre choix et explication :	8
3.1.3. Principe de fonctionnement de la méthode de l'extraction des visage d'« opencv » :	8
3.1.4. L'architecture du modèle :	9
3.1.5. Test sur notre base de données :	9
3.2. Reconnaissance faciale :	11
3.2.1 Résultats comparatifs :	14
3.2.2 VGG-Face :	14
3.3. Combinaison des deux algorithmes et fonctionnement :	15
4. Partie Hardware et implémentation :	16
5. Difficultés rencontrées :	17
6. Propositions pour futur travail :	17
7. Résumé pour le fonctionnement final et la solution qu'on a implémentée :	18
8. Avantages de cette solution et points forts :	18
9. Conclusion :	18
10. Ressources :	19
11. Annexe	20

1. Introduction :

Dans le cadre de notre projet thématique de cette année, nous avons décidé de travailler sur une innovation dans les services administratifs. Ce projet consiste à développer un dispositif de reconnaissance des étudiants qui permettra d'établir une liste de présence et de l'envoyer au service administratif concerné. Il ne sera plus nécessaire de confier une feuille de présence au délégué pour la faire circuler parmi les étudiants de la promotion, tout se fera automatiquement et de manière autonome.

1.1. Définition du projet :

La conception d'un système de reconnaissance faciale en temps réel pour remplacer la feuille d'émargement dans une salle de cours ou d'examen. Ce système utilise les techniques d'apprentissage profond (Deep Learning) pour identifier les étudiants présents dans la salle à partir de la prise quotidienne des photos par une caméra.

1.2. Cahier de charges :

Pour la réalisation de ce projet, on a commencé par la définition des objectifs et la faisabilité de ce projet, et on a suivi le cahier des charges suivant :

Fonctionnement principale :
Un système qui contient une base de données des étudiants d'une classe.
Ce système va prendre une image au minimum une fois chaque 30 minutes des étudiants dans une salle.
Le système va traiter cette image et extraire (détecter) des visages dans l'image.
Le système va traiter chaque visage et il va faire la reconnaissance faciale de chaque visage.
Il va remplir un tableau avec les élèves présents dans la salle.
Il va enregistrer toutes les informations tel que l'heure de la prise de l'image, la date et la salle et la classe.
Il va envoyer ces informations sur internet.
Cas étranges :
Lorsqu'il y a un élève qui n'est pas de la même classe, le système va détecter cette personne comme inconnue.
Le système va enregistrer l'image de visage de la personne inconnue ainsi de l'image initiale complète qui contient tous les visages.
Un dossier avec la date du jour et qui contient tous ces fichiers va être envoyé sur internet.

Figure 01 : Tableau de cahier de charge.

1.3. Le matériel nécessaire pour la réalisation de ce projet :

- Une carte Raspberry PI avec son alimentation et une carte mémoire.
- Un module caméra.
- Un pc ordinateur pour le développement de l'application et les tests.
- Un point d'accès internet pour la connexion de la carte avec internet.

2. Stratégie de travail :

Après l'étude de notre cahier de charge, on a décidé de partager le travail en deux parties, une partie de développement informatique (partie Software) et une partie de développement système embarqué (partie Hardware).

Pour la partie de développement software on a suivi la stratégie suivante :

- On commence par des recherches sur les méthodes qui existe pour la détection et l'extraction des visages et ainsi des modèles pour la reconnaissance faciale basées sur le traitement d'images et des méthodes basées sur l'intelligence artificielle.
- Au même temps on collecte quotidiennement des images pour nos collègues pour l'étude et le teste de notre système.
- Après qu'on trouve des méthodes, on fera des tests sur des images téléchargé d'internet et ensuite on fera une étude comparative entre les méthodes pour sortir la meilleure méthode qui satisfait notre cahier de charge.
- Après qu'on trouve la méthode ou le modèle IA et le valider sur des images sur internet, on teste notre solution sur notre base de données collecté.
- Après les tests, on cherche à valider le modèle sur notre base de données. Si le modèle ne fonctionne pas comme attendu, on doit changer le modèle ou chercher d'autres solutions et répéter les étapes précédentes.

Pour la partie de développement Hardware on a suivi la stratégie suivante :

- On fait la commande du matériel.
- On prépare l'environnement de programmation pour le développement.
- On fait l'implémentation de notre solution dans la carte.
- On fait des tests sur des images.
- On connecte le système avec un module caméra et on effectue des tests et en temps réel.
- On développe sur la carte la partie des cas étranges et l'envoi des données.

2.1. Répartition des taches :

On est deux personnes qui travaille dans ce projet,

Iheb BOUARICHE, va prendre la partie de la recherche et de développement de la solution basée sur l'intelligence artificielle pour la reconnaissance faciale, son rôle est de réalisé un programme qui permet de prendre une image à l'entrée et sortir une liste avec les personnes présentes dans cette image. Il va utiliser les techniques de traitement d'images et de l'apprentissage profond.

Arslane MEGHELLI, va être responsable du développement Hardware, il va réaliser un système embarqué capable de prendre une image en temps réel (toute les 30 minutes maximum) et de la transmettre au programme développé par Iheb. Ce système devra ensuite extraire la liste des étudiants présents et d'autres informations, les enregistrer dans un fichier Excel, et envoyer ces données sur internet.

2.2. Méthodologie de travail :

Chaque semaine, on a un jour dédié au partage de l'avancement de notre travail, ainsi qu'aux recherches, solutions et propositions qu'on a trouvées. Ensuite, on définit les points pour le travail à venir. À chaque fois qu'on développe une solution, on l'implémente sur la carte pour en vérifier la faisabilité. On échange des idées et on adapte notre stratégie chaque fois qu'on rencontre un blocage ou une situation compliquée.

2.3. Changement de stratégie :

Avant, on avait décidé de développer un système pour effectuer l'apprentissage sur un grand nombre d'images de nos collègues. Cependant, on a réalisé que cela n'était pas faisable lorsque de nouveaux étudiants rejoignent la classe pour deux raisons principales :

- D'abord, il n'est pas possible de demander à un étudiant de fournir un grand nombre de ses photos.
- Ensuite, il est compliqué d'apprendre un modèle d'intelligence artificielle à chaque fois qu'un nouvel étudiant est ajouté à la base de données. Ceci est d'autant plus problématique étant donné que notre système sera implémenté sur une carte Raspberry, un petit ordinateur avec des limitations en termes de puissance de calcul.

2.4. Solution validée :

Après des recherches et des tests de plusieurs solutions, on a finalement, validé l'idée de développer un système qui prend une seule photo d'un nouvel élève, et l'ajouter dans la base de données. Ensuite et en mode RUN, à partir de cette image on peut faire la reconnaissance de son visage dans tous les images prise par le système embarqué en temps réel.

3. Fonctionnement de la partie Software :

Cette partie contient deux éléments essentiels : la détection et l'extraction des visages dans une image contenant plusieurs visages, ainsi que la reconnaissance faciale.

Dans ce rapport, on va expliquer le fonctionnement de chaque élément.

3.2. Extraction et détection des visages :

C'est la localisation des visages dans une image. Après une recherche approfondie sur internet, on a trouvé qu'il existe plusieurs méthodes pour l'extraction des visages dans une image qui contient plusieurs visages.

3.1.1. Teste pour différentes méthodes :




































Notre problème c'était de prendre ces méthodes et tester ses performances pour faire le meilleur choix.

On a pris des images sur internet et on a testé ces méthodes sur différentes images comme vous montre le tableau suivant.

On a pris comme paramètres la précision et la vitesse d'exécution (temps d'exécution du programme de l'entrée jusqu'à l'obtention des visages). Voici l'étude comparative des différentes méthodes :



Figure 02 : l'une des images de test pour le test des méthodes d'extraction des visages.

Modèle	Résultats										
opencv	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
											
	Temps de détection 1.390409231185913 seconds										
ssd	F1	F2									
											
	Temps de détection 0.13123464584350586 seconds										
mtcnn	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
											
	Temps de détection 2.428948163986206 seconds										
retinaface	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
											
	Temps de détection 8.324228763580322 seconds										




Yolov8	F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 
	Temps de détection 0.33027172088623047 seconds
yunet	F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 
	Temps de détection 0.06933712959289551 seconds
fastmtcnn	F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12 
	Temps de détection 1.2057855129241943 seconds

Figure 03 : résultats de chaque modèle de l'extraction de visage, ainsi que le temps d'exécution.

Commentaire : On remarque qu'il y a plusieurs méthodes qui détectent tous les visages dans l'image, et il y en a qui en détectent moins et d'autres qui se trompent dans la détection.

On a réalisé une étude sur plusieurs autres images contenant plusieurs visages, et on a analysé le temps pris par chaque modèle pour la détection. Ensuite, on a calculé la moyenne du temps d'exécution pour chaque modèle sur une machine locale. Voici le graphe obtenu :

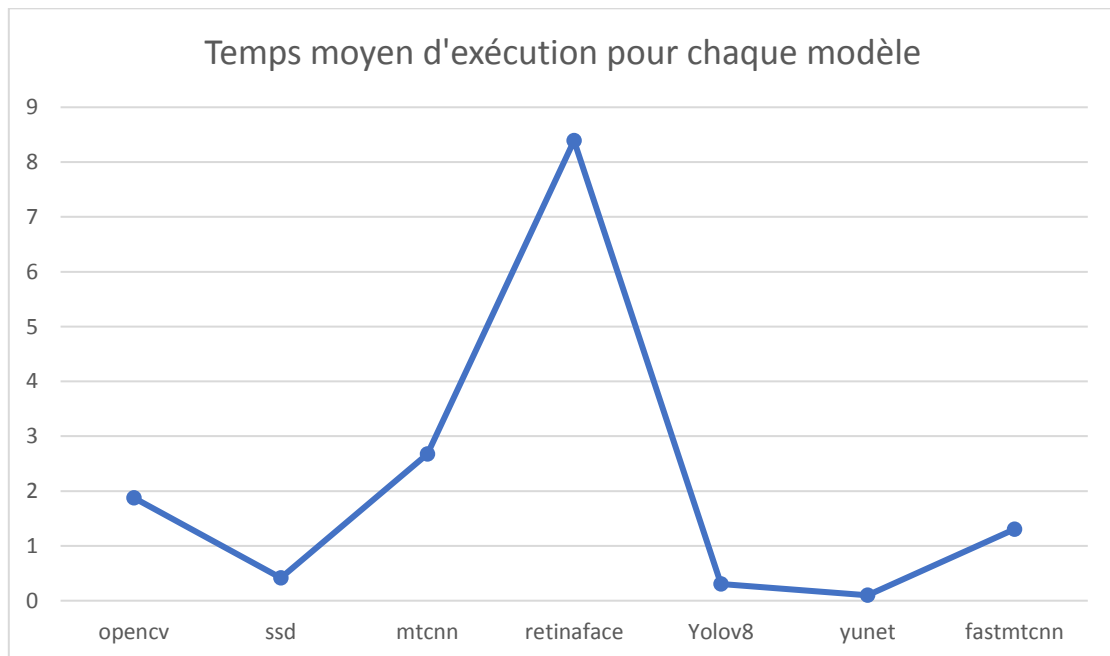


Figure 04 : graphe pour le temps moyen d'exécution pour chaque méthode de détection.

Commentaire : On voit que y a des méthodes qui prennent un temps important pour l'exécution alors que y en a d'autres qui sont rapides en exécution.

3.1.2. Notre choix et explication :

Après plusieurs tests, on a choisi le modèle d'extraction de visage d'OpenCV. Ce choix s'est fondé sur le fait que ce modèle détecte uniquement la position d'un rectangle autour de chaque visage dans une image. En comparaison, les autres méthodes appliquent des transformations à l'image, telles que l'alignement 3D et la coupe qui ne montre pas la totalité des visages. Cette décision va jouer un rôle important pour la reconnaissance faciale, car pour obtenir une précision en reconnaissance, il est essentiel de conserver un maximum de caractéristiques faciales.

3.1.4. Principe de fonctionnement de la méthode de l'extraction des visage d'« opencv » :

Maintenant puisqu'on a choisi la méthode d'OpenCV pour la détection des visages, on a cherché et compris cette méthode à partir des articles de recherche et des sources sur le web [1]. On va détailler un peu son fonctionnement par la suite.

Cette méthode est basée sur des techniques de Machine Learning pour la détection des visages, elle est développée en passant par trois étapes suivantes :

- **Le calcul de l'image intégrale** : la méthode consiste à sortir les caractéristiques dans l'image en utilisant des filtres, la méthode de l'image intégrale va nous permettre de calculer ces caractéristiques rapidement en échelle et en position. On va sortir les caractéristiques de HAAR pour cette application.
Les caractéristiques de HAAR : sont des filtres de deux dimensions et rectangulaire, ces filtres sont utilisés pour l'identification des structures dans une image (tel que les lignes, les textures, etc...), il existe plusieurs types des filtres de HAAR : filtres de deux rectangles, de trois et de quatre rectangles. Ensuite et après l'application de ces filtres sur les images en position et en échelle, on obtient un nombre très important de caractéristiques, ce qui peut être lourd en calcul pour un modèle de classification.
- **L'algorithme AdaBoost** : Cet algorithme est utilisé pour sélectionner les caractéristiques les plus importantes pour la classification entre une image contenant un visage et une autre sans visage. AdaBoost est un classificateur qui combine plusieurs classificateurs entraînés pour créer un classificateur puissant.
 - **Création des Classificateurs** : AdaBoost combine ces caractéristiques sélectionnées pour former des classificateurs faibles. Chaque classificateur faible est un modèle simple qui est légèrement meilleur que le hasard pour distinguer entre les visages et les non-visages. AdaBoost attribue des poids aux exemples d'entraînement, en augmentant les poids des exemples mal classés afin que les classificateurs faibles suivants se concentrent davantage sur ces exemples difficiles.
 - **Construction d'un Classificateur Fort** : En combinant plusieurs classificateurs faibles pondérés, AdaBoost construit un classificateur fort. Ce classificateur fort est une combinaison linéaire des classificateurs faibles, où chaque classificateur faible contribue en fonction de son poids déterminé par AdaBoost. On dit qu'un classificateur AdaBoost plus fort s'il prend plus de caractéristiques et donc il a plus de classificateur faible.
- **Classification en cascade** : La troisième méthode consiste à réaliser des étages en cascade. Chaque étage contient un classificateur AdaBoost et, à chaque étage, on trouve un modèle plus fort. Cette technique est utilisée pour éliminer, à chaque étage, les régions qui ne contiennent pas de visages, ce qui permet d'augmenter la rapidité du calcul. Il faut comprendre que, pour qu'une région soit identifiée comme contenant un visage, elle doit passer avec succès tous les étages.

3.1.4. L'architecture du modèle :

Ce modèle utilise 38 couches (étages) de classificateurs AdaBoost, entraînés sur une base de données de 4916 images de visages labellisées avec une résolution de 24 x 24, ainsi que 9544 images ne contenant pas de visages. Ces images comprennent un total de 350 millions de petites images de 24 x 24.



Figure 05 : exemple des images des visages utilisé pour l'apprentissage des classificateurs.

3.1.5. Test sur notre base de données :

Après le test sur une image de notre base de données de nos camarades dans une salle, on a obtenu les résultats suivants :



Figure 06 : l'image à l'entrée de système de détection.



Figure 07 : Les images des visages détectés après l'exécution du code de détection.

Commentaire : on remarque bien que le programme a détecté tous les visages qui sont dans l'image à l'entrée (à condition que les visages doivent être positionnés en face la caméra).

Après l'obtention des visages, on va développer une solution qui prend chaque visage et le traiter pour le reconnaître.

3.2. Reconnaissance faciale :

Après des recherches sur les méthodes qui peuvent être utilisées pour la reconnaissance faciale, on a trouvé plusieurs approches, telles que :

- Entraîner un classificateur d'images : Cette méthode nécessite une collecte de données importante, donc on l'a annulé.
- Utiliser un vecteur caractéristique (Embedding Vector): Cette méthode consiste à prendre une image, et ensuite extraire un vecteur caractéristique d'un visage, puis calculer la distance avec d'autres visages. Si la distance est inférieure à un certain seuil, cela signifie que c'est la même personne dans l'image. Cette méthode a bien fonctionné et c'est celle qu'on va utiliser pour notre solution.

L'avantage de cette méthode est qu'il suffit de prendre une seule photo d'un nouvel élève pour l'ajouter à la base de données.

Pour cette solution, on a testé différents modèles de Deep Learning pré-entraînés sur notre base de données.

Le programme prend deux images contenant chacune un visage et les passe, l'une après l'autre, dans un réseau de neurones profond (modèle de l'IA). À la sortie, on obtient un vecteur caractéristique de chaque image (en anglais : "embedding vector"). Le programme compare ensuite la distance euclidienne entre les deux vecteurs à un certain seuil. Si la distance est inférieure au seuil, les deux images représentent la même personne. Sinon, ce n'est pas la même personne. Voici les résultats pour quelques tests qu'on a fait sur les images de nos camarades :



Figure 08 : Deux images de la même personne (avec et sans barbe)

Et voici le tableau avec les différents modèles

Le modèle	Distance	Temps d'exécution	La même personne ?	Seuil
VGG-Face	0.6192805123478301	2.4109108448028564	Oui	0.68
Facenet	0.33399397091011995	1.6701862812042236	Oui	0.4
Facenet512	0.32227831143130603	1.640101671218872	Non	0.3
OpenFace	0.26733561871553846	1.201770305633545	Non	0.1
DeepFace	0.21537111581274848	5.225113153457642	Oui	0.23
DeepID	0.047543969387073504	1.4541990756988525	Non	0.015
ArcFace	0.4843091708996228	1.629307746887207	Oui	0.68
SFace	0.5325555920972895	1.0362541675567627	Oui	0.593

Figure 09 : Tableau de résultat de reconnaissance faciale.

Commentaire : On peut voir que les méthodes VGG-Face, Facenet, DeepFace, ArcFace et SFace ont donné la bonne prédiction, alors que les autres n'ont pas réussi à détecter que c'était la même personne.



Figure 10 : Deux images de la même personne.

Dans cette deuxième expérience, on va tester les modèles sur deux images de la même personne mais avec un éclairage et des expressions faciales différentes. Voici les résultats :

Le modèle	Distance	Temps d'exécution	La prédiction	Seuil
VGG-Face	0.6736759059017616	2.937410831451416	True	0.68
Facenet	0.6236478723951333	2.9040067195892334	False	0.4
Facenet512	0.4236998679687729	2.8820929527282715	False	0.3
OpenFace	0.3712229096442299	2.053914785385132	False	0.1
DeepFace	0.23026103484438232	5.59130859375	False	0.23
DeepID	0.12204879577616157	2.554736375808716	False	0.015
ArcFace	0.37433675889583307	2.63346791267395	True	0.68
SFace	0.24810227303972132	1.9013197422027588	True	0.593

Figure 11 : Tableau de résultat de reconnaissance faciale.

Commentaire : On peut clairement voir que seuls les modèles VGG-Face, ArcFace et SFace ont donné le bon résultat, tandis que les autres ont échoué.

Remarque : On ajoute que ArcFace et SFace ne sont pas précis lorsqu'il y a un changement d'âge ou de style de la personne, comme on peut le voir dans l'exemple suivant :



Figure 12 : Deux images de la même personne.

Maintenant, pour cette image, on va tester les modèles de la même personne avec une photo simple et l'autre avec une casquette et des lunettes.

Le modele	Distance	Temps d'exécution	la prediction	Seuil
VGG-Face	0.6133414794757963	3.248588800430298	True	0.68
Facenet	0.38187203233407196	1.4804351329803467	True	0.4
Facenet512	0.37281731632954174	1.4907350540161133	False	0.3
OpenFace	0.30112648391917396	0.9992396831512451	False	0.1
DeepFace	0.38139448891818506	5.371493816375732	False	0.23
DeepID	0.1470279082756102	0.7740786075592041	False	0.015
ArcFace	0.6186461820371727	1.947218656539917	True	0.68
SFace	0.7803979012523027	1.0754146575927734	False	0.593

Figure 13 : Tableau de résultat de reconnaissance faciale.

Commentaire : On peut voir maintenant que les modèles VGG-Face, Facenet et ArcFace ont donnés les bonnes prédictions.

Par la suite, on a effectué une étude comparative basée sur un article de recherche, analysant différents modèles avec testé avec des bases de données très grandes. On va argumenter notre choix en utilisant également ces résultats.

3.2.1. Résultats comparatifs :

TABLE IV
THE ACCURACY OF DIFFERENT METHODS EVALUATED ON THE LFW DATASET.

Method	Public. Time	Loss	Architecture	Number of Networks	Training Set	Accuracy±Std(%)
DeepFace [20]	2014	softmax	Alexnet	3	Facebook (4.4M,4K)	97.35±0.25
DeepID2 [21]	2014	contrastive loss	Alexnet	25	CelebFaces+ (0.2M,10K)	99.15±0.13
DeepID3 [36]	2015	contrastive loss	VGGNet-10	50	CelebFaces+ (0.2M,10K)	99.53±0.10
FaceNet [38]	2015	triplet loss	GoogleNet-24	1	Google (500M,10M)	99.63±0.09
Baidu [58]	2015	triplet loss	CNN-9	10	Baidu (1.2M,18K)	99.77
VGGface [37]	2015	triplet loss	VGGNet-16	1	VGGface (2.6M,2.6K)	98.95
light-CNN [85]	2015	softmax	light CNN	1	MS-Celeb-1M (8.4M,100K)	98.8
Center Loss [101]	2016	center loss	Lenet+7	1	CASIA-WebFace, CACD2000, Celebrity+ (0.7M,17K)	99.28
L-softmax [104]	2016	L-softmax	VGGNet-18	1	CASIA-WebFace (0.49M,10K)	98.71
Range Loss [82]	2016	range loss	VGGNet-16	1	MS-Celeb-1M, CASIA-WebFace (5M,100K)	99.52
L2-softmax [109]	2017	L2-softmax	ResNet-101	1	MS-Celeb-1M (3.7M,58K)	99.78
Normface [110]	2017	contrastive loss	ResNet-28	1	CASIA-WebFace (0.49M,10K)	99.19
CoCo loss [112]	2017	CoCo loss	-	1	MS-Celeb-1M (3M,80K)	99.86
vMF loss [115]	2017	vMF loss	ResNet-27	1	MS-Celeb-1M (4.6M,60K)	99.58
Marginal Loss [116]	2017	marginal loss	ResNet-27	1	MS-Celeb-1M (4M,80K)	99.48
SphereFace [84]	2017	A-softmax	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.42
CCL [113]	2018	center invariant loss	ResNet-27	1	CASIA-WebFace (0.49M,10K)	99.12
AMS loss [105]	2018	AMS loss	ResNet-20	1	CASIA-WebFace (0.49M,10K)	99.12
Cosface [107]	2018	cosface	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.33
Arcface [106]	2018	arcface	ResNet-100	1	MS-Celeb-1M (3.8M,85K)	99.83
Ring loss [117]	2018	Ring loss	ResNet-64	1	MS-Celeb-1M (3.5M,31K)	99.50

Figure 14 : Tableau pour étude comparative des différents modèles de reconnaissance faciale.

Ce tableau représente une étude comparative des différents modèles de reconnaissance faciale. On constate que le modèle VGGface contient 16 couches et utilise un seul réseau, atteignant une précision de 98.8%. Comparé aux autres modèles, certains utilisent des réseaux de neurones de taille importante, ce qui rend leur exécution lente. De plus, le modèle VGG-Face est entraîné sur une base de données comportant un grand nombre de visages différents (classes), ainsi qu'une grande variation de position, d'âge, d'expression et de conditions d'éclairage, ce qui le rend robuste.

Le modèle VGG-Face a obtenu une précision de 98.95% avec la base de données LFW (contenant plus de 13 000 images de visages de 5 749 personnes différentes, prises dans des conditions non contrôlées). De même, il a atteint une précision de 98.95% avec la base de données YTF (comprenant 3 425 vidéos de 1 595 personnes, offrant une grande variabilité dans les poses, les expressions faciales, les conditions d'éclairage et les arrière-plans).

Après plusieurs expériences, on a choisi le modèle VGG-Face. Ce choix est basé sur les résultats obtenus des articles de recherche et sur nos propres tests sur notre base de données. On en conclure que le VGG-Face est le plus robuste par rapport à tous les autres modèles. Par exemple, dans un cas difficile de reconnaissance avec des conditions d'éclairage et des expressions très différentes, le modèle VGG-Face a réussi à détecter la même personne.

3.2.2. VGG-Face :

Le modèle utilisé pour VGG-Face est basé sur une variante de l'architecture VGG16, qui est un modèle de réseau de neurones convolutif (CNN).

➤ Son Architecture :

- Le modèle VGG-Face suit l'architecture VGG16, qui se compose de 16 couches.
- Les couches incluent des couches convolutives, des couches entièrement connectées.
- Le réseau se compose typiquement de 13 couches convolutives, 5 couches de max-pooling et 3 couches entièrement connectées.

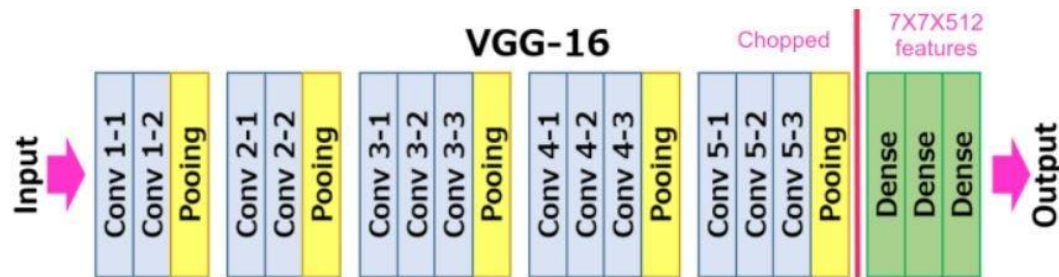


Figure 15 : l'architecture de VGG16.

➤ Entraînement :

- Le modèle VGG-Face a été initialement entraîné sur le dataset VGGFace, qui contient 2,6 millions d'images de 2 622 célébrités.
- L'entraînement implique l'utilisation d'un dataset à grande échelle pour capturer une large gamme de variations des visages humains.

On ajoute qu'on a remarqué que la puissance de VGG-Face viennent de la base de données qui est très large et aussi qui contient une gamme de variation très grande, tel que l'âge, l'éclairage et le style etc...

3.3. Combinaison des deux algorithmes et fonctionnement :

Après avoir testé plusieurs combinaisons d'algorithmes de détection et de reconnaissance, on a obtenu des résultats différents. La meilleure combinaison a été d'utiliser le modèle OpenCV pour détecter les visages et les extraire comme des images, puis de faire passer ces images dans un modèle VGG-Face. À la sortie, on obtient un vecteur qui sera comparé avec les vecteurs des étudiants de la classe pour déterminer s'il s'agit d'un élève présent ou non.

Les autres algorithmes de détection n'ont pas donné de bonnes performances en raison de la modification des images de visages et de l'absence de recadrage direct. De plus, le modèle VGG-Face est entraîné sur des images qui ne sont pas alignées et qui présentent différentes poses. C'est pourquoi la combinaison de VGG-Face avec le modèle OpenCV s'est avérée être la meilleure solution.

4. Partie Hardware et implémentation :

Maintenant, on a le modèle de détection et de reconnaissance testé et validé sur différentes bases de données. Notre objectif est d'implémenter ce modèle dans un système embarqué pour qu'il soit fonctionnel en temps réel dans une salle de cours.

On a utilisé une carte Raspberry Pi pour cette application en raison de sa capacité en mémoire et en puissance de calcul. On a installé un système sur cette carte et préparé un environnement de développement Python avec les bibliothèques nécessaires.

Ensuite, on a implémenté le programme de reconnaissance sur la carte et effectué des tests d'exécution.

Puis, on a développé la solution complète :

- Prise d'une image toutes les 10 minutes.
- Détection et reconnaissance faciale sur cette image.
- Génération de la liste de présence.
- Sauvegarde de cette liste dans un fichier Excel avec les informations nécessaires.
- Stockage du fichier Excel, des images des visages inconnus et des images de toute la classe (prises par la caméra au début) dans un dossier, puis transmission sur Internet.

On a terminé la partie de développement matériel et, après des tests et validations, le système est maintenant opérationnel.

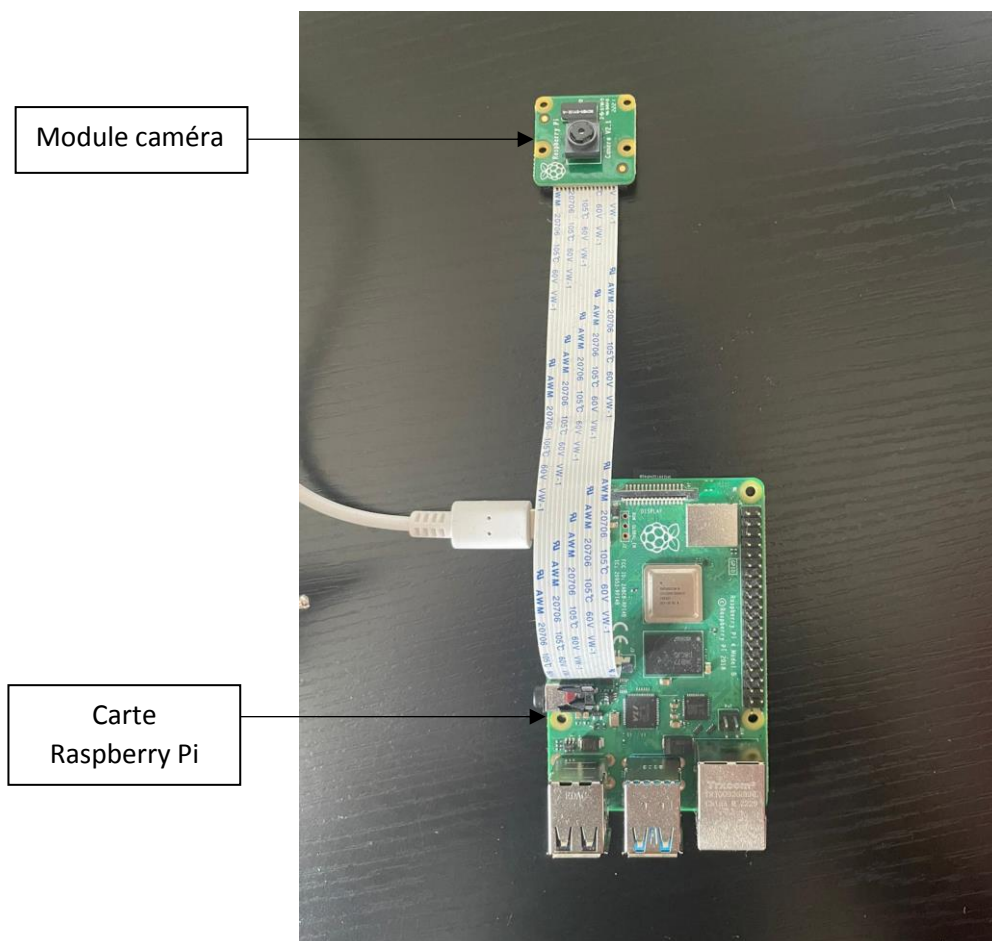


Figure 16 : Le système embarqué (carte Raspberry + Camera)

5. Difficultés rencontrées :

Pour la partie logicielle, on a rencontré un problème pour la collecte de données, notamment la prise quotidienne de photos. Nos camarades de classe n'ont pas accepté cette idée, donc on a réalisé cette partie avec nos amis hors de la classe. Cependant, il était difficile de les réunir dans une classe chaque jour. On a alors demandé leurs photos individuelles et créé une base de données virtuelle, en effectuant des modifications pour simuler les visages dans la classe. Vous pouvez voir cela dans les deux images suivantes : une réelle et une autre virtuelle, cette base de données était utilisée pour le test et la validation finale de notre système de détection et de reconnaissance.



Figure 17 : image à gauche (virtuel) – image à droite réelle.

Commentaire : Vous pouvez voir qu'on a remplacé le visage d'une personne par celui d'une autre dans l'image à gauche.

La deuxième difficulté qu'on a rencontrée concerne la commande du matériel. On a suivi les consignes et envoyé la liste du matériel nécessaire, mais on n'a pas reçu le matériel. On a donc commandé le matériel par nos propres moyens.

6. Propositions pour futur travail :

- Connexion de ce système avec l'emploi du temps, la salle et la classe pour que le système soit programmé automatiquement pour la prise des images, en déterminant le moment précis pour chaque prise et en fournissant plus d'informations sur la base de données en cours de traiter.
- Amélioration du système par la collecte quotidienne des images des élèves. De cette façon, on obtiendra une base de données très grande, permettant de réaliser des tests et d'améliorer les performances de notre système.
- Connexion de ce système avec la plateforme ENT, et donner l'accès aux élèves pour qu'ils puissent consulter leurs absences.
- Développement d'un système qui calcule automatiquement le coefficient d'absence.
- Développement d'un système automatique qui envoie un email directement à l'élève absent, lui demandant de justifier son absence.

7. Résumé pour le fonctionnement final et la solution qu'on a implémentée :

Notre solution actuelle n'est pas très différente de notre cahier des charges, et notre objectif est resté le même. On a réussi à développer un système qui prend une photo toutes les 10 minutes, détecte et reconnaît les visages présents sur les images, et les enregistre comme présents dans une liste de présence avec l'heure et la date exactes.

Si le système détecte une personne inconnue, il enregistre l'image de son visage. Ensuite, il envoie un dossier contenant la liste de présence dans un fichier Excel, les images des visages inconnus, et l'image complète de toute la classe sur Internet.

8. Avantages de cette solution et points forts :

- Numérisation et automatisation de la feuille d'émargement.
- Garantir la précision et fiabilité, car notre système va nous donner la présence de chaque élève chaque 10 minutes.
- Sécurité et surveillance, car le système peut détecter les personnes qui ne sont pas de la même classe.
- En cas d'incident, les images capturées peuvent servir de preuves pour une investigation.
- Facilement adaptable à différentes tailles de classes et à des environnements variés.
- Réduit l'utilisation de papier et les tâches administratives liées à la gestion manuelle des présences.

9. Conclusion :

Le développement et l'implémentation d'un système de reconnaissance faciale en temps réel pour la gestion de la présence dans une salle de classe représentent une avancée significative vers l'automatisation et la modernisation des environnements éducatifs. En utilisant des techniques d'apprentissage profond et des algorithmes sophistiqués pour la détection et la reconnaissance des visages, ce système offre de nombreux avantages, notamment une grande précision, une gestion automatisée des présences, une sécurité renforcée, et une meilleure gestion des données.

L'intégration de ce système sur une plateforme embarquée telle qu'une carte Raspberry Pi démontre également la faisabilité de cette technologie avec des ressources limitées, tout en assurant une exécution en temps réel. Les défis rencontrés, comme la collecte de données et la gestion des nouveaux élèves, ont été surmontés grâce à des solutions innovantes, assurant ainsi la robustesse et l'efficacité du système.

En conclusion, ce projet non seulement améliore l'efficacité administrative et pédagogique, mais il pose également les bases pour d'autres applications potentielles de la reconnaissance faciale dans divers domaines, faisant progresser la technologie dans la vie quotidienne.

10. Ressources :

Les articles de VGG-FACE :

[1] <https://arxiv.org/pdf/1710.08092>

[2] <https://arxiv.org/pdf/1409.1556>

[3] <https://arxiv.org/pdf/1804.06655>

L'article pour la cascade classifier :

[4] <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>

Les ressources et sites WEB :

[5] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[6] <https://viso.ai/computer-vision/deepface/>

11. Annexe :

Le code commenté :

```
from deepface import DeepFace
import matplotlib.pyplot as plt
import time
import numpy as np
import subprocess
import cv2
import pandas as pd
import datetime
import pytz
import os

#Cette partie on l'active(=1) que pour le debuggage
Debug = 0
k = 1
#On lance une boucle infini pour que le programme tourne tant que le système est allumé
while 1:
    start_time = time.time() #Le temps en début du programme
    command = "sudo libcamera-still -o test.jpg" #la commande à lancer dans le cmd, cette
    commande est pour connecter le système avec le module caméra et prendre une photo.
    subprocess.run(command, shell=True) #Exécution de la commande

    # Charger l'image avec OpenCV
    image = cv2.imread("test.jpg")

    # Vérifie si l'image a été chargée correctement
    if image is not None:
        if Debug:
            # Affiche l'image
            cv2.imshow("Output Image", image)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
        else:
            #Si y a une erreur, on affiche ce message
            print("Erreur: Impossible de charger l'image.")

    inputimage = image

    # Partie de détection des visages
    if Debug:
        print('Face detection modele => fastmtcnn')
    img = DeepFace.extract_faces(inputimage, detector_backend='opencv',
enforce_detection=False)
    if len(img) < 1:
        continue
```



```

if len(img) > 0:
    for j, face_data in enumerate(img):
        face_array = face_data['face']
        if Debug:
            plt.subplot(1, len(img), j + 1)
            plt.imshow(face_array)
            plt.axis("off")
            plt.title(f"F{j+1}")

# Partie de reconnaissance faciale
Dict1 = {1: "DATA/Amine/06.jpeg", 2: "DATA/Anes/3.jpeg", 3:
"DATA/Arselane/03.jpeg", 4: "DATA/Brahim/Reference.jpeg", 5: "DATA/Iheb/08.jpg", 6:
"DATA/Rimy/Reference.jpeg"}
#Dict2 = {1: "DATA/Amine/02.jpeg", 2: "DATA/Anes/2.jpeg", 3:
"DATA/Arselane/02.jpeg", 4: "DATA/Brahim/02.jpeg", 5: "DATA/Iheb/02.jpeg", 6:
"DATA/Rimy/02.jpeg"}
Dict = {1: "Amine", 2: "Anes", 3: "Arselan", 4: "Brahim", 5: "Iheb", 6: "Rimy"}

if Debug:
    print("Face recognition model => Facenet")
ListDePresence = []
ListeDeDistance = []
Repitition = []
v = 0

for j, face_data in enumerate(img):
    if Debug:

print("#####")
#####)
        face_array = face_data['face']
        for i in range(6):
            image_array_scaled = (face_array * 255).astype(np.uint8)
            result = DeepFace.verify(Dict1[i+1], image_array_scaled,
model_name="VGG-Face", detector_backend="opencv", enforce_detection=False)
            if Debug:
                print(result)
            #L'ajout des visages reconnus dans une liste
            if result["verified"] == True:
                v = v + 1
                if v < 2:
                    ListDePresence.append(Dict[i+1])
                    ListeDeDistance.append(result["distance"])
                if v == 2:
                    ListDePresence.pop(-1)
                    ListDePresence.append("Inconnu")
                    ListeDeDistance.pop(-1)

```

```

        ListeDeDistance.append("Inconnu")
    if v == 0:
        ListDePresence.append("Inconnu")
    v = 0
if Debug:
    print("Temps de detection", execution_time, "seconds")
    print(ListDePresence)
    print(ListeDeDistance)
    print(v)
l = 1
ListDePresence_sans_inconnus = []
for j, face_data in enumerate(img):
    if ListDePresence[j] == "Inconnu":
        face_array = face_data['face']
        filename = f"inconnu{l}.png"
        plt.imsave(filename, face_array)
        if Debug:
            print(f"Saved face image: {filename}")
        l = l + 1
    else:
        ListDePresence_sans_inconnus.append(ListDePresence[j])

#Dans la partie suivante on va enregistrer les résultats dans un fichier Excel avec
d'autres informations tel que la date
paris_timezone = pytz.timezone('Europe/Paris')
current_time_paris = datetime.datetime.now(paris_timezone).replace(tzinfo=None)

# Vérification si le fichier existe
def file_exists(filepath):
    return os.path.isfile(filepath)
file_path = "/home/pi/Desktop/presence.xlsx"
if file_exists(file_path):
    # la lecture du fichier Excel
    df_existing = pd.read_excel(file_path, engine='openpyxl')
else:
    # Sinon on fait la création d'un nouveau fichier Excel
    df_existing = pd.DataFrame(columns=['Name.'+str(k), 'Presence.'+str(k),
'Date.'+str(k)])

# Creation de la base de donnée a partir de dictionnaire
df_new = pd.DataFrame.from_dict(Dict, orient='index', columns=['Name.'+str(k)])

# ajouter la column de presence et de la date
df_new['Presence.'+str(k)] = df_new['Name.'+str(k)].apply(lambda x: 'present' if x in
ListDePresence_sans_inconnus else 'absent')
df_new['Date.'+str(k)] = current_time_paris

if not df_existing.empty:

```

```

df_new.index = df_existing.index

# Concatenation avec la base de donnée existante
df_combined = pd.concat([df_existing, df_new[['Name.'+str(k), 'Presence.'+str(k),
'Date.'+str(k)]]], axis=1)

df_combined.to_excel(file_path, index=False, engine='openpyxl')

# afficher tout les résultats dans le fichier excel
print(df_combined)
k = k+1
start_time = time.time()
#Partager le fichier Excel sur internet
command = "sudo cp /home/pi/Desktop/presence.xlsx /var/www/html/"
subprocess.run(command, shell=True)

#Attente de 10 min pour que le programme se relance
elapsed_time = time.time() - start_time
sleep_time = max(0, 600 - elapsed_time)
time.sleep(sleep_time)

```