INSTITUT NATIONAL DES SCIENCES APPLIQUÉES CENTRE VAL DE LOIRE

THANH DUY KHONG

Rapport CONFIDENTIEL
(cocher la case correspondante)

☒ NON
☐ **OUI**

Si oui est coché, signature du tuteur entreprise obligatoire

5A GSI - Option ACAD

Année Universitaire :

2022 / 2023

Analyse d'engagement d'émotion d'élèves sur des videos en classe en utilisant Deep Learning

RAPPORT DE STAGE

Tuteur Laboratoire :

PEJMAN Rasti

Enseignant / Chercheur

Enseignant référent :

HIDANE Moncef

L'ESAIP

ANGERS 49000

esaip La Salle
ÉCOLE D'INGÉNIEURS

Remerciements

Je tiens à exprimer ma sincère gratitude envers toutes les personnes qui ont contribué à mon parcours de stage et à l'achèvement de ce rapport. Bien que mon temps chez l'ESAIP d'Angers ait été relativement court, le soutien, les conseils et les expériences que j'ai acquis sont inestimables pour ma croissance personnelle et professionnelle.

Tout d'abord, je suis extrêmement reconnaissant envers Monsieur Pejman RASTI, mon superviseur, pour m'avoir offert l'opportunité de faire partie de ce projet. Votre mentorat, votre patience et votre volonté de partager vos connaissances ont été essentiels pour façonner ma compréhension du domaine de Deep Learning. Je suis reconnaissant pour les retours constructifs que vous m'avez fournis et pour m'avoir confié des tâches qui ont contribué à mon apprentissage.

J'étends mes remerciements à mon collègue Hugo Voyneau pour la convivialité et la coopération dans le processus de travail en commun.

Je suis redevable envers mon établissement scolaire, l'INSA Centre Val de Loire, d'avoir facilité cette opportunité de stage et d'avoir transmis les connaissances qui ont posé les bases de mes contributions pendant cette période. Le lien entre mon apprentissage académique et son application pratique chez l'ESAIP a été éclairant.

Enfin, je souhaite exprimer ma gratitude envers toute autre personne qui, directement ou indirectement, a joué un rôle dans la création d'une expérience de stage significative.

Ce rapport est le reflet des efforts collectifs de toutes les personnes qui m'ont guidé, encadré et partagé leurs connaissances avec moi. Merci d'avoir été une partie intégrante de mon parcours de stage.

Résume

Les émotions jouent un rôle très important dans la vie humaine, affectant directement l'efficacité du travail. Dans l'environnement d'apprentissage, les émotions peuvent jouer un rôle important en soutenant ou en sapant l'apprentissage et l'enseignement. Comprenant l'importance des émotions dans l'enseignement et l'apprentissage, la faculté des lettres de l'IUT d'Angers Université a collaboré avec Deparment en intelligence artificielle et Big Data de l'ESAIP pour créer le projet NeuroCam - le projet sur lequel j'ai travaillé pendant mon stage. L'objectif de ce projet était de créer une méthode d'analyser automatiquement les émotions de chaque élève après chaque leçon à partir des données recueillies au cours de cette leçon (vidéo et données d'oxymètres). Cela aide les titulaires de classe à ajuster les méthodes et le contenu de manière appropriée pour améliorer la qualité de l'enseignement ainsi que la capacité des élèves à acquérir des connaissances.

Mon travail principal pendant le stage a été de collecter des données réelles avec des volontaires, de lire des rapports scientifiques, de convertir des données en données numériques, d'analyser des données numériques à l'aide de méthodes scientifiques, d'apprendre et d'identifier l'émotion dominante de chaque personne dans chaque classe et l'évolution des émotions de cette personne au cours d'une semestre. Au cours du processus de mise en œuvre, j'ai rencontré de nombreuses difficultés en raison de la complexité du problème, les données réelles sont souvent incomplètes et sous-optimales, il y avait donc souvent beaucoup de d'erreur à corriger.

Après 6 mois de stage, j'ai construit une méthode complète pour analyser les émotions des étudiants conformément aux exigences initiales.

Mots-clés : Analyse d'émotion dans classe, face-detection, python, pytorch, opencv, FaceNet,Traitement d'image, KMeans

Abstract

Emotions play a very important role in human life, directly affecting work efficiency. In the learning environment, emotions can play an important role in supporting or undermining learning and teaching. Understanding the importance of emotions in teaching and learning, the Faculty of Letters of the IUT of Angers University collaborated with the Artificial Intelligence and Big Data Department of ESAIP to create the NeuroCam project - the project on which I worked during my internship. The objective of this project was to create a method to automatically analyze the emotions of each student after each lesson from the data collected during this lesson (video and oximeter data). This helps classroom teachers adjust methods and content appropriately to improve the quality of teaching as well as students' ability to acquire knowledge.

My main work during the internship were to collect real data with volunteers, to read scientific reports, to convert data into numerical data, to analyze numerical data using scientific methods, to learn and identify the dominant emotion of each person in each class and the evolution of that person's emotions over the course of a semester. During the implementation process, I encountered many difficulties due to the complexity of the problem. The actual data is often incomplete and suboptimal, so there was often a lot of error to correct.

After 6 months of internship, I built a complete method to analyze students' emotions according to the initial requirements.

Keywords: Emotion analysis in classe, face-detection , python, pytorch, opencv, FaceNet, Image Processing , KMeans

Sommaire

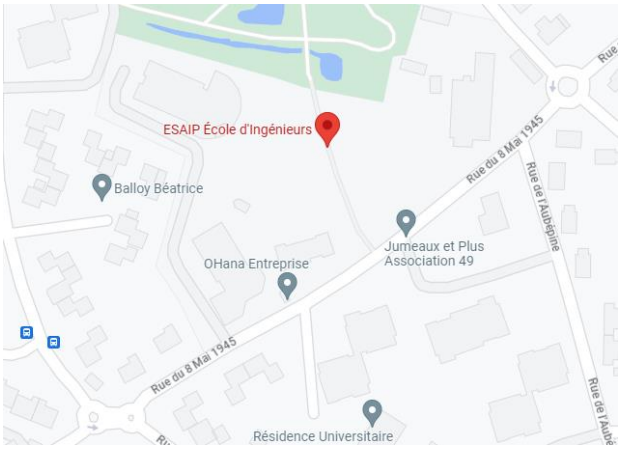
I. Introduction

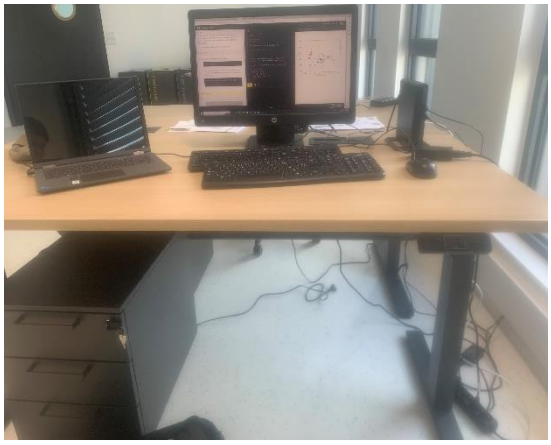
1.1 Informations générales sur l'ESAIP

Établissement d'Enseignement Supérieur Privé d'Intérêt Général (EESPIG), l'Esaip est une école d'ingénieurs lassallienne créée en 1988, habilitée par la Commission des Titres d'Ingénieur. L'école fait également partie de la Conférence des Grandes Écoles et de la Fesic. L'Esaip forme des ingénieurs de haut niveau scientifique et technique, en Numérique et en Gestion des Risques et Environnement, deux spécialités au cœur d'enjeux civilisationnels.

La spécialisation se fait dès le post bac. Le mode projet et la pédagogie active sont favorisés pour faire de chacun de nos étudiants un acteur et un contributeur de sa formation. Il s'agit d'enseigner et d'apprendre autrement, plus efficacement, avec plus d'interactivité, afin de développer l'implication et l'autonomie des élèves, de prendre en compte les diversités et les évolutions, de favoriser le multiculturalisme. Les résultats sont là : notre taux d'insertion professionnelle est de 100 %, avec, en moyenne, un temps de recherche d'emploi inférieur à 3 semaines et un premier salaire à 37 000€ brut.

À l'ESAIP Angers, J'appartiens au département de recherche de Big Data et IA. Mon lieu de travail régulière est la salle des stagiaires, bâtiment C, 1er étage.





1.2 Problématique

L'émotion a une influence substantielle sur les processus cognitifs chez l'homme, y compris la perception, l'attention, l'apprentissage, la mémoire, le raisonnement et la résolution de problèmes. L'émotion a une influence particulièrement forte sur l'attention, notamment en modulant la sélectivité de l'attention ainsi qu'en motivant l'action et le comportement. De nombreux rapports indiquent que les émotions positives améliorent non seulement la réceptivité des élèves, mais augmentent également la mémoire à long terme. Les expériences d'apprentissage peuvent combiner les capacités émotionnelles et le développement scolaire améliorera la qualité de l'environnement d'apprentissage dans son ensemble

Cependant, capturer les vrais sentiments de tous les élèves n'est pas facile pour les enseignants, surtout lorsqu'une classe compte beaucoup d'élèves. La collecte de données par le biais de la méthode de remplissage de formulaires n'est pas non plus vraiment fiable, car les étudiants craignent de donner des évaluations négatives sur des enseignants. Ainsi, il est nécessaire d'avoir une méthode d'évaluation entièrement automatique, précise et facile à appliquer afin que les enseignants puissent évaluer les sentiments des élèves après chaque leçon, afin que les enseignants puissent avoir des changements adaptés à ses méthodes d'enseignement.

1.3 Objectifs et buts du stage

La tâche principale de ce stage est de développer des méthodes simples pour identifier les émotions des élèves en classe grâce à des données recueillies auprès de bénévoles.

(1) Conception, mise en œuvre et installation d'équipements d'enregistrement à faible coût dans les salles de classe.

(2) Concevoir et mettre en œuvre une base de données numériques à partir de de données brutes

(3) Construire une méthode pour prédire les émotions basées sur les données de l'oxymètre

(4). L'objectif principal est de développer une méthode d'analyse des données en utilisant les données de la vidéo et des oxymètres. **Notez qu'il ne s'agit pas d'une reconnaissance faciale directe des émotions dans la vidéo, mais de construire une méthode analyse à partir des données combinées de la vidéo et de l'oxymètre de données dans une leçon entière.**

(5) Si le temps le permet, développement de la visualisation en temps réel et des rapports aux enseignants

Le stagiaire est attendu du lundi au vendredi, de 9h à 17, soit 35 heures par semaine. Il est important que les stagiaires respectent ces horaires afin de s'assurer de la disponibilité et de la continuité de leur travail au sein de l'organisation.

Ce stage demande des connaissances de base en Python, Machine Learning, une familiarité avec les librairies de Deep Learning telles que Torch, Tensroflow .... C'est un stage très pertinent avec les connaissances que j'ai apprises en 5ème année ainsi que mon orientation professionnelle qui est de devenir un développeur ML.

Ce stage débute le 6 février et se termine le 6 août, soit 24 semaines.

Annonce du plan de rapport :

II. Développement

2.1 Recherche complémentaire sur les connaissances nécessaires au projet

Durant le premier mois de mon stage, j'ai lu des rapports scientifiques pour compléter mes connaissances en traitement d'images et en bibliothèques python pour travailler avec des photos et des vidéos. De nos jours, il y avait centaines de méthodes pour détecter les visages humaines .Les trois méthodes suivantes sont les trois principales méthodes d'identification d'un visage humain dans une image fixe :

Viola-Jones algorithme

L'algorithme Viola Jones porte le nom de deux chercheurs en vision par ordinateur qui ont proposé la méthode en 2001, Paul Viola et Michael Jones dans leur article, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Bien qu'il soit assez obsolète, Viola Algorithm est toujours l'un des cadres populaires en raison de son efficacité et de son faible coût de calcul.

Avant d'en savoir plus sur l'algorithme de Viola Jones, nous devons comprendre le concept Haar – likes features (Haar - caractéristiques). Les caractéristiques de Haar sont une séquence de fonctions de forme carrée redimensionnées proposées par Alfred Haar en 1909. Elles sont similaires aux noyaux de convolution enseignés dans le cours Convolution Neural Networks. Nous appliquerons ces caractéristiques Haar à toutes les parties pertinentes du visage afin de détecter le visage humain.

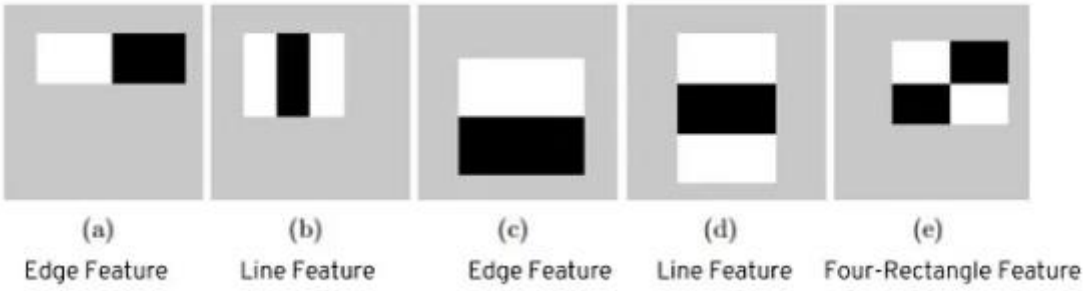
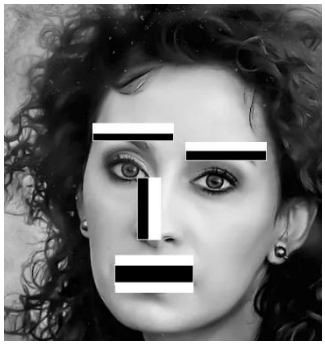


Fig.1 : 5 principales caractéristiques *de Haar*

Tous les visages humains partagent certaines propriétés universelles du visage humain, comme la région des yeux est plus sombre que ses pixels voisins et la région du nez est plus lumineuse que la région des yeux. Ces caractéristiques sur l'image permettent de trouver facilement les bords ou les lignes de l'image, ou de sélectionner des zones où il y a un changement soudain dans les intensités des pixels.

Comme nous l'avons vu dans l'exemple ci-dessus, Haar - caractéristiques peuvent être utilisés pour identifier les traits du visage. Les caractéristiques de bord (1) sont utilisées pour détecter les sourcils



parce que le front et les sourcils forment une zone de plus clairs - plus sombres image. De même, pour détecter les lèvres, nous utilisons une fonction similaire à celle de Haar (image (3)) avec des

Fig.2 : Caractéristiques de Haar appliquées sur les parties pertinentes du visage

pixels plus clairs, plus sombres et plus clairs. Pour détecter le nez, nous pourrions utiliser la fonction de type Haar plus sombre-plus claire de (image (1)). Et ainsi de suite.

Ainsi, afin de localiser le visage, nous devons déterminer l'emplacement des traits de type Haar dans l'images. Selon l'algorithme Viola-Jonas, pour détecter la caractéristique de type Haar présente dans une image, la formule ci-dessous doit donner un résultat plus proche de 1. Plus la valeur est proche de 1, plus le changement de détection de la caractéristique Haar dans l'image est important.

$$\Delta = zone\ noir - zone\ blanche = \frac{1}{n_{noir}} \sum_{1}^{n_{noir}} I(x) - \frac{1}{n_{blanche}} \sum_{1}^{n_{blanche}} I(x)$$

La fonction Δ calcule la différence entre la moyenne des valeurs de pixel dans la région plus sombre et la moyenne des valeurs de pixel dans la région plus claire. Dans le meilleur cas, tous les pixels de la zone noire ont une valeur d'un et les zones blanches ont une valeur de 0.

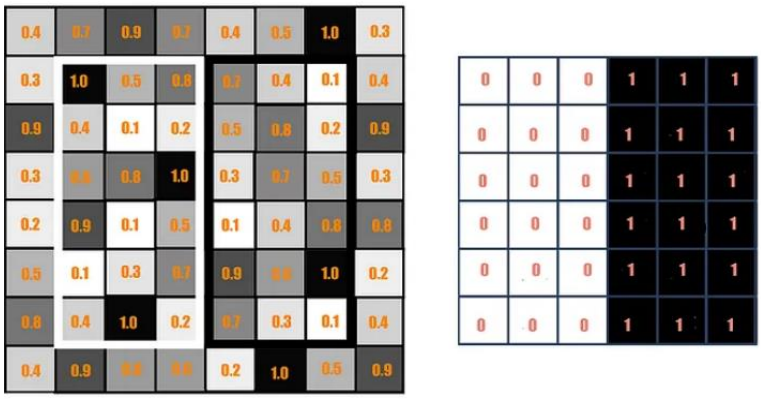


Fig.3 : Exemple de calcul de Haar entre zones claires et sombres

Dans l'exemple de gauche, les valeurs de haar entre les zones sombres et claires sont de 0.51 – 0.53 = -0,02. Cela signifie qu'il n'y a pas les caractéristiques de Haar entre les deux zones.

Pour détecter des caractéristiques de Haar n'importe où dans l'image, la fonction Haar doit traverser toute l'image dans les directions de gauche à droite et de haut en bas, similaire à la couche de convolution. De plus, toutes les tailles possibles des caractéristiques de Haar seront appliquées

La traversée des caractéristiques Haar sur une image impliquerait beaucoup de calculs mathématiques parce que nous devons recalculer la somme de tous les pixels du rectangle à chaque fois que nous glissons. En réalité, ces calculs peuvent être très intensifs puisque le nombre de pixels serait beaucoup plus important lorsqu'il s'agit d'un grand filtre de caractéristiques. Pour résoudre ce problème, ils ont introduit une méthode appelée Intégrale Image. Une image intégrale est calculée à partir de l'image d'origine de telle sorte que chaque pixel de celle-ci soit la somme de tous les pixels situés à sa gauche et au-dessus dans l'image d'origine

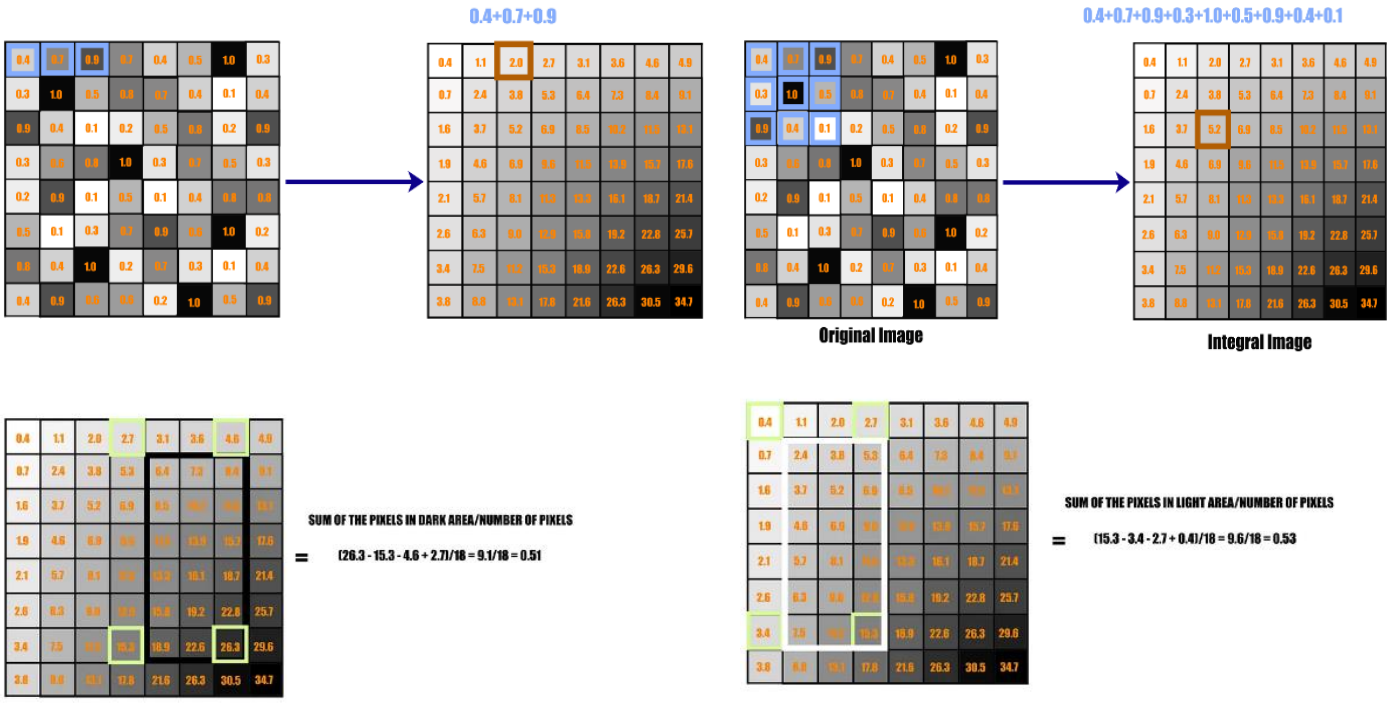
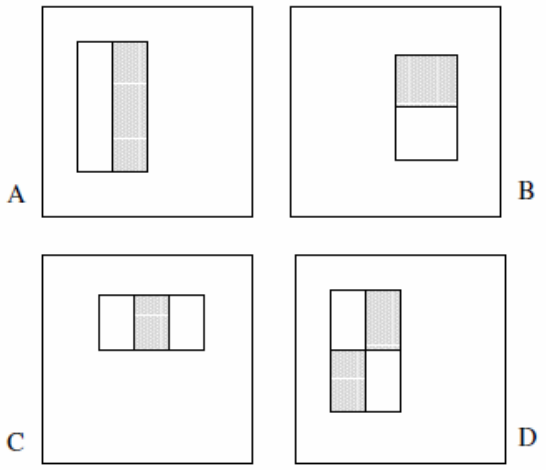


Fig. 4 : Exemple d' efficacité de la méthode intégrale de l' image .

Grâce à cette méthode, nous n'avons besoin de calculer l'image intégrale qu'une seule fois par image. L'image reste à la même taille et la valeur de pixel moyenne dans la zone rectangulaire est également préservée. Avec l'image intégrale, seuls 4 ajouts de valeur constante sont nécessaires à chaque fois pour n'importe quelle taille d'entité (par rapport aux 18 ajouts précédents). Cela réduit progressivement la complexité temporelle de chaque ajout, car le nombre d'ajouts ne dépend plus du nombre de pixels inclus.

Viola Jones algorithme repose sur le placement d'un sous-frame de 24x24 pixels dans une image. Les sous-frames coulissent de gauche à droite et de haut en bas. Dans chaque sous-frame,nous allons ensuite placer des filtres rectangulaires à l'intérieur dans toutes les positions avec toutes les tailles possibles (1x4, 1x6, 1x8, ..., 1x24, 2x2, 2x4, 2x6, 2x8, ..., 2x24, 3x2, 3x4, 3x6, ..., jusqu'à 24x24) pour détecter tous les caractéristiques de Haar possible

Les mathématiciens estiment qu'il existe un total d'environ 180 000 filtre de caractéristiques de Haar possibles pour un sous-frame de 24 x 24. La majorité de ces filtres ne fonctionneront pas bien ou ne



seront pas pertinentes pour les traits du visage, car elles seront trop aléatoires pour trouver quoi que ce soit. Donc, ici, ils avaient besoin d'une technique de sélection de fonctionnalités, pour sélectionner

un sous-ensemble de fonctionnalités parmi le vaste ensemble qui non seulement sélectionnerait des fonctionnalités plus performantes que les autres, mais éliminerait également celles qui ne sont pas pertinentes. Ils ont utilisé une technique de renforcement appelée AdaBoost, dans laquelle chacune de ces 180 000 fonctionnalités a été appliquée aux images séparément pour créer des apprenants faibles. Certains d'entre eux produisaient de faibles taux d'erreur car ils séparaient mieux les images positives des images négatives que les autres, tandis que d'autres ne le faisaient pas. Ces apprenants faibles sont conçus de manière à ne mal classer qu'un nombre minimum d'images. Ils peuvent être plus performants qu'une supposition aléatoire. Avec cette technique, leur ensemble final de Caractéristiques a été réduit à un total de 6000 d'entre eux

Le sous-ensemble de toutes les 6000 Caractéristiques s'exécutera à nouveau sur les images d'entraînement pour détecter s'il y a une caractéristique faciale présente ou non. Au lieu d'avoir à calculer 6000 fonctionnalités pour chaque fenêtre, nous allons le décomposer en petites étapes, Les étapes au début contiennent des filtres le plus simples. Le traitement de la deuxième étape ne commencerait que lorsque les caractéristiques de la première étape seraient détectées dans l'image. Au contraire, si l'étape initiale ne détecte rien sur la fenêtre, supprimez la fenêtre elle-même du processus restant et passez à la fenêtre suivante. Le nombre de fonctionnalités dans les cinq premières étapes est de 1, 10, 25, 25 et 50, et cela a augmenté dans les étapes suivantes. Les étapes initiales avec un nombre plus simple et moindre de fonctionnalités ont supprimé la plupart des fenêtres sans caractéristiques faciales, réduisant ainsi le taux de faux négatifs, tandis que les étapes ultérieures avec un nombre complexe et plus de fonctionnalités peuvent se concentrer sur la réduction du taux de détection d'erreurs, réalisant ainsi faible taux de faux positifs

Il y a au total 38 étapes définies pour la méthode Viola Jonas. En fonction de la taille des fenêtres coulissantes et de l'emplacement du visage, du nombre de caractéristiques, le visage peut être détecté à une certaine étape.

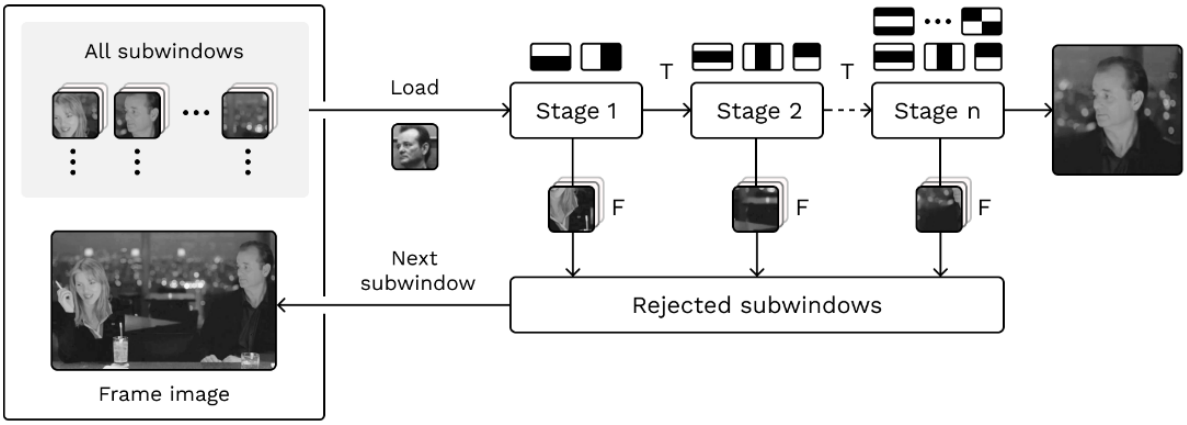


Fig. 5 : Illustration le processus de travail de Haar Cascade Classifier

Le processus ci-dessus s'appelle Haar Cascade Classifier. En raison de sa complexité, dans ce projet, j'utilise sa bibliothèque intégrée pour identifier les visages dans les vidéos.

Modèle d'apprentissage en profondeur

Après avoir déterminé la position du visage dans la vidéo, nous devons également trouver l'emplacement de la visage et la comparer avec une image originale de cette personne. Actuellement, il existe de nombreux modèles d'apprentissage en profondeur utilisés pour comparer les visages. En général, les modèles d'apprentissage en profondeur pour la reconnaissance faciale sont construits à l'aide de réseaux de neurones convolutifs (CNN), qui sont un type de réseau de neurones conçu pour traiter des données structurées en grille, telles que des images.

Certains modèles populaires peuvent être mentionnés comme : VGGFace/VGGFace2 , FaceNet , DeepFace , OpenFace , Multi-task Cascaded Convolutional Networks (MTCNN) ….. Dans ce projet, nous utiliserons la modèle FaceNet.

FaceNet est un réseau neuronal profond utilisé pour extraire des caractéristiques d'une image du visage d'une personne. Il a été publié en 2015 par les chercheurs de Google Schroff et al

FaceNet prend une image du visage de la personne en entrée et produit un vecteur de 128 nombres qui représentent les caractéristiques les plus importantes d'un visage. En DeepLearning, ce vecteur est appelé embedding. Toutes les informations importantes d'une image sont intégrées dans ce vecteur.

La principale différence entre FaceNet et d'autres techniques est qu'il apprend le mappage à partir des images et crée des embedding plutôt que d'utiliser une couche de goulot d'étranglement pour les tâches de reconnaissance ou de vérification. Une fois les embeddings créées, toutes les autres tâches telles que la vérification, la reconnaissance, etc. peuvent être effectuées à l'aide de techniques standard de ce domaine particulier, en utilisant ces embedding nouvellement générées comme vecteur de caractéristiques. Par exemple, nous pouvons utiliser k-NN pour la reconnaissance faciale en utilisant des embedding comme vecteur de caractéristiques et de la même manière, nous pouvons utiliser n'importe quelle technique de regroupement pour regrouper les visages ensemble et pour la vérification, nous avons juste besoin de définir une valeur seuil (threshold valeur)

L'entraine de FaceNet utilise un mécanisme appelé Triple Loss (Perte de triplet) pour sa fonction de perte (loss function). L'intuition derrière la fonction de perte de triplet est que nous voulons que notre image d'ancrage (image d'une personne spécifique A) soit plus proche des images positives (toutes les images de la personne A) par rapport aux images négatives (toutes les autres images)

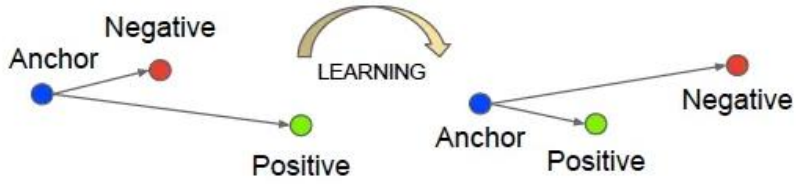


Fig.6 : Illustration simple du fonctionnement de la méthode Triple Loss

Un exemple "dur positif" fait référence à une paire d'images où les deux images représentent le même individu (c'est-à-dire qu'elles ont le visage de la même personne), mais elles peuvent être difficiles à faire correspondre en raison de variations d'éclairage, de pose, d'expression ou d'autres facteurs. En fournissant des exemples positifs, le modèle apprend à mieux gérer ces variations et à affiner son processus d'extraction de caractéristiques pour reconnaître avec précision les visages dans différentes conditions.

Un exemple "négatif dur" fait référence à une paire d'images où les images sont d'individus différents (c'est-à-dire qu'elles sont de visages de personnes différentes), mais le modèle peut initialement les confondre comme similaires en raison de certaines caractéristiques visuelles partagées. La formation avec des exemples négatifs durs aide le modèle à améliorer sa capacité à différencier les visages similaires de différentes personnes. Ceci est important pour réduire les fausses correspondances positives dans la reconnaissance faciale

La fonction de perte de triplet peut être formellement définie comme :

$$L = \max\left(\left[\left\|f(x_i^a) - f(x_i^p)\right\|^2 - \|f(x_i^a) - f(x_i^n)\|^2 + \alpha\right], 0\right)$$

x_i *représenter une image*

$f(x_i)$ *représente embedding d'une image*

α *représentent est un hyperparamètre qui représente la séparation minimale souhaitée entre*

$les\ paires\ positives\ et\ négatives. Il\ empêche\ le\ réseau\ de\ trop\ effondrer\ les\ embeddings$

Ici, les exposants a, p et n correspondent respectivement aux images d'ancrage, positives et négatives.

Pendant la l'entrainement, le modèle est présenté avec des lots d'images de triplets. Il calcule les embeddings des images d'ancrage, positives et négatives, puis calcule la perte de triplet en utilisant les distances entre ces plongements. Les paramètres du modèle (pondérations et biais) sont mis à jour pour minimiser la perte de triplet en utilisant l'optimisation de descente de gradient AdaGrad

L'entraînement avec la perte de triplet aide le modèle à apprendre les intégrations qui capturent efficacement les relations inhérentes entre des images similaires et différentes, ce qui facilite la discrimination entre différents individus ou objets dans l'espace d'intégration. Ceci est particulièrement utile dans des tâches telles que la reconnaissance faciale, où il est important de s'assurer que les visages similaires sont rapprochés tandis que les visages différents sont écartés dans l'espace des fonctions apprises.

Les détails du processus de l'etrainement sont beaucoup plus compliqués et je ne peux pas tous les couvrir ici. FaceNet utilise 2 types de CNN, à savoir l'architecture Zeiler & Fergus et le modèle Inception de style GoogLeNet

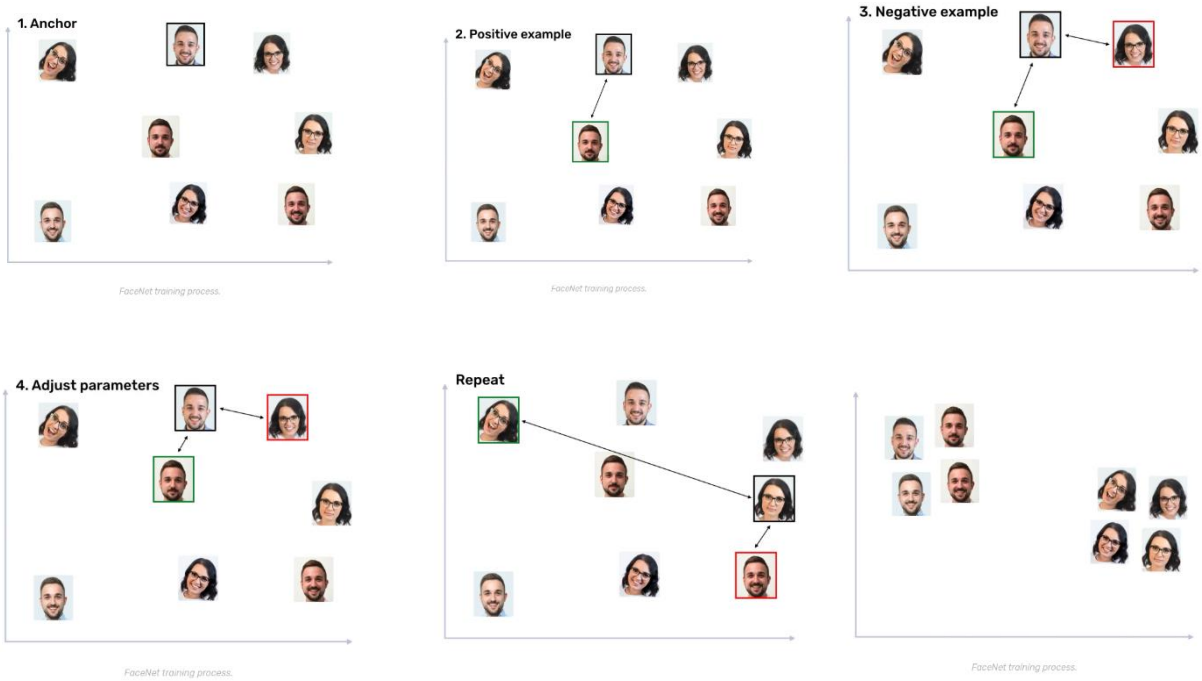


Fig. 7 Un exemple de la façon dont Facenet a appris à trouver des photos de la même personne

2.2 Mettre en pratique les connaissances acquises sur les supports disponibles

J'ai pratiqué la reconnaissance faciale et appliqué la bibliothèque FER (Facial emotion recognition) pour identifier les émotions directement en temps réel sur les données de l'année dernière

Fig. 8 : Le modèle FER reconnaît les émotions en temps réel

2.3 Construire la base de données

2.3.1 Collecte de bases de données en classe

Toutes les données utilisées dans ce projet sont des données réelles collectées par moi à la faculté de Langues de l'Université d'IUT d'Angers. Lors de notre première rencontre avec la direction du projet, nous avons mentionné l'inefficacité de la construction d'une base de données pour former un modèle CNN pour la classifier des émotions dans les vidéos pour les raisons suivantes :

1) Le modèle CNN a besoin d'une quantité très importante et variée de données d'images pour être précis. Dans le monde réel, il y a parfois des étudiants qui gardent toujours un visage fixe, peu importe le type d'émotions qu'ils ont.
2) Construire une base d'étiquettes pour les données des trains est également un grand problème. Dans le projet de l'année dernier (2022), après chaque leçon, chaque volontaire recevait une enquête, dans laquelle chaque personne cochait ses sentiments toutes les 5 minutes. Cependant, la majorité des résultats sont neutres, car en partie ils ne se souviennent pas correctement de leurs émotions au fil du temps ou ne les reconnaissent pas correctement. Cette étiquette fait que le modèle prédit toujours l'émotion sur l'échantillon de test comme neutre

Par conséquent, nous avons proposé d'utiliser une méthode plus efficace et précise, qui consiste à utiliser une méthode d'apprentissage non supervisée. Au lieu d'utiliser une feuille de suivi, nous utiliserons un autre outil appelé oxymètre.

Un oxymètre se compose généralement d'un petit appareil en forme de clip qui est généralement attaché au bout du doigt, au lobe de l'oreille ou à l'orteil d'une personne. À l'intérieur de l'oxymètre, il y a des diodes électroluminescentes (DEL) et un détecteur de lumière. L'appareil fonctionne sur la base d'un principe appelé spectrophotométrie. L'oxymètre émet deux types de lumière : généralement une lumière rouge et une lumière infrarouge. Ces lumières traversent le tissu (généralement le bout du doigt) où l'oxymètre est fixé. De l'autre côté du tissu, un détecteur de lumière mesure la quantité de lumière qui traverse ou est absorbée par le tissu

Le sang oxygéné et désoxygéné absorbe la lumière différemment. L'hémoglobine, la protéine du sang qui transporte l'oxygène, absorbe plus de lumière infrarouge et laisse passer plus de lumière rouge lorsqu'elle transporte de l'oxygène. D'autre part, l'hémoglobine désoxygénée absorbe plus de lumière rouge et laisse passer plus de lumière infrarouge. L'Oxymètre utilise les deux types de lumière pour mesurer la quantité de pulsations sanguines au bout du doigt. Cette pulsation est causée par la rythme cardiaque. L'oxymètre calcule la fréquence cardiaque en analysant ces impulsions et en comptant combien de fois votre cœur bat en une minute. Il affiche ensuite votre fréquence cardiaque sur son écran.

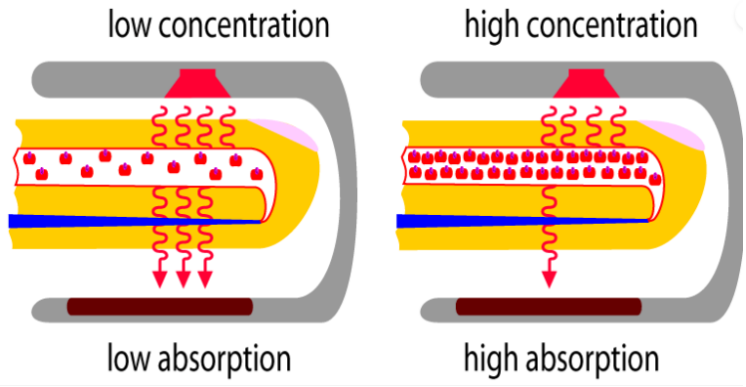


Fig.9 : Illustration du fonctionnement d'un oxymètre

L'émotion est étroitement liée à la fréquence cardiaque, nous avons donc choisi la variation de la fréquence cardiaque comme indicateur pour pouvoir classer les émotions à l'aide de la méthode K-

Means. Dans le même temps, l'oxymètre peut répondre aux exigences d'une installation facile et bon marché dans la salle de classe, évitant de déranger les étudiants.

La collecte de donnés a eu lieu dans un cours de français pour les étrangers. La plupart d'entre eux sont au niveau débutant et parlent très peu le français, la communication entre professeurs et élèves est donc assez difficile. C'est aussi la raison pour laquelle les deux écoles ont collaboré pour créer ce projet, dans l'espoir qu'il puisse aider les enseignants à capter les émotions des élèves même lorsqu'il n'y a pas besoin de communiquer.

Nous avons sélectionné 5 volontaires pour participer au projet. Ces 5 personnes viennent de nombreux pays différents, sont en bonne santé et n'ont pas de problèmes cardiaques. La collecte des données commence du 8 mars au 16 avril, tous les mercredis (salle A117 de 9h30 à 11h), jeudis (salle L209 de 14h40 à 16h40) et vendredis (salle A018 de 9h30 à 12h10). Toutes les activités d'apprentissage se déroulent naturellement



Fig. 10 : Vidéo de bénévoles dans 3 classes différentes

Les outils utilisés pour la collecte comprennent : Une caméra haute définition KIYO PRO, 5 oxymètre Wellue , 1 trépied pour régler la hauteur de la caméra , des cables . J'ai également écrit un code qui change la résolution de la vidéo en 720p. La caméra peut capturer des visages en haute définition même si les élèves sont assis loin de la caméra ou dispersés (voir la 3ème image ci-dessus).


```python
fps = 24.0
res = '720p'
record_seconds= 30
# Set resolution for the video capture
def change_res(cap, width, height):
    cap.set(3, width)
    cap.set(4, height)

# Standard Video Dimensions Sizes
STD_DIMENSIONS = {
    "480p": (640, 480),
    "720p": (1280, 720),
    "1080p": (1920, 1080),
    "4k": (3840, 2160),
}


# grab resolution dimensions and set video capture to it.
def get_dims(cap, res='720p'):
    width, height = STD_DIMENSIONS["1080p"]
    if res in STD_DIMENSIONS:
        width,height = STD_DIMENSIONS[res]
    ## change the current caputre device
    ## to the resulting resolution
    change_res(cap, width, height)
    return width, height

# Video Encoding, might require additional installs
VIDEO_TYPE = {
    'avi': cv2.VideoWriter_fourcc(*'XVID'),
    #'mp4': cv2.VideoWriter_fourcc(*'H264'),
    'mp4': cv2.VideoWriter_fourcc(*'XVID'),
}

def get_video_type(filename):
    filename, ext = os.path.splitext(filename)
    if ext in VIDEO_TYPE:
      return  VIDEO_TYPE[ext]
    return VIDEO_TYPE['mp4']


cap = cv2.VideoCapture(0)
out = cv2.VideoWriter(filename, get_video_type(filename), 25, get_dims(cap, res))
```

Fig. 11:Outils utilisés pour collecter les données (à gauche) et le code de la caméra (droite)

Avant chaque cours, chaque élève portera un oxymètre au doigt, qui sera ensuite connecté à son smartphone via l'application téléphonique dédiée de l'oxymètre appelée ViHealth. Cette application enregistrera vos données de fréquence cardiaque dans un rapport sous forme de graphique. Après chaque cours, les étudiants partageront ce rapport avec moi par courriel

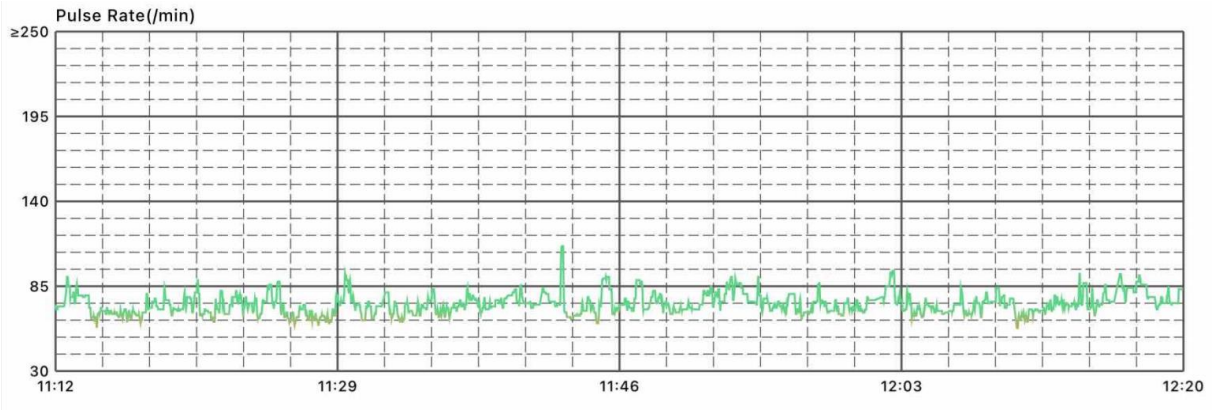


Fig. 12 : Données brutes de l'oxymètre sous forme de graphiques obtenus

Nous avons cherché des solutions pour pouvoir collecter des données chaque seconde, mais la plupart des applications mobiles ne produisent également que des données graphiques au lieu de données numériques. Il ne m'est pas possible d'écrire une telle application par moi-même en si peu de temps, de plus, nous n'avons pas besoin de trop nous préoccuper des moments où la fréquence cardiaque est normale, mais seulement des moments où il y a un changement remarquable, ce qui indique un changement dans les émotions.

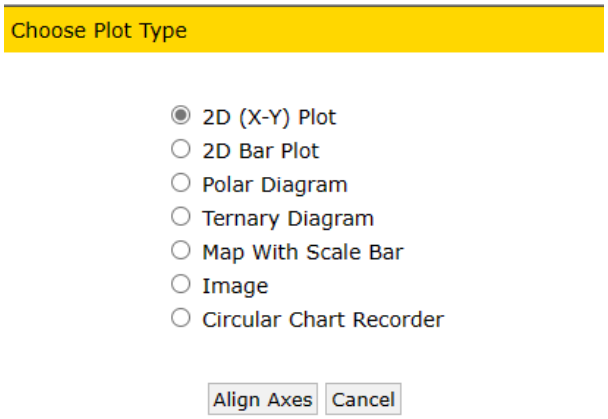
A la fin de chaque cours, mon collègue Hugo (du département de psychologie de l'IUT universitaire) distribuera des fiches d'évaluation générale de la leçon, à partir desquelles il analysera les principales émotions des étudiants lors de cette séance. Je vais utiliser ces données pour comparer avec mes résultats

2.3.2 Traitement de données brut

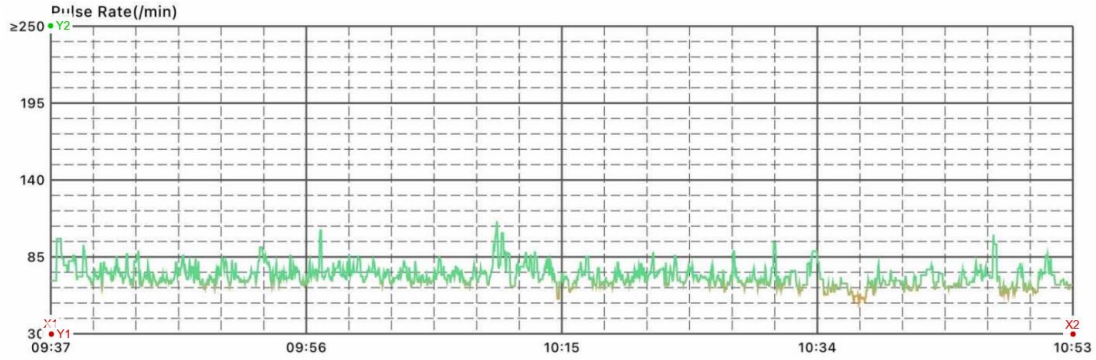
2.3.2.1 Convertir des données d'oxymètre en forme numerique

Après avoir collecté les données, le premier travail consiste à convertir les données d'image en données numériques. Écrire un programme pour le faire efficacement est trop difficile pour moi. Par conséquent, après avoir discuté avec le tuteur, j'ai décidé d'utiliser un logiciel intégré sur le site Web appelé **WebPlotDigitizer.** Il s'agit d'un logiciel gratuit dédié à la conversion des données d'image sous forme métrique. Notre travail ne nécessite que les étapes suivantes :

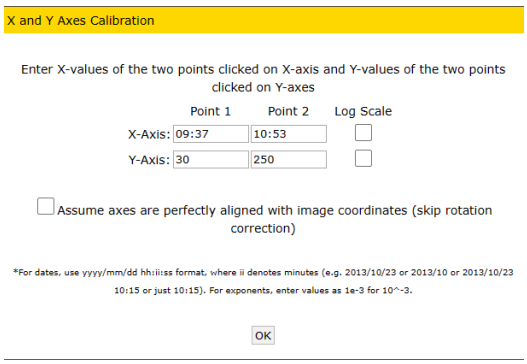
1) Charger le graphique et choisir un type de graphique



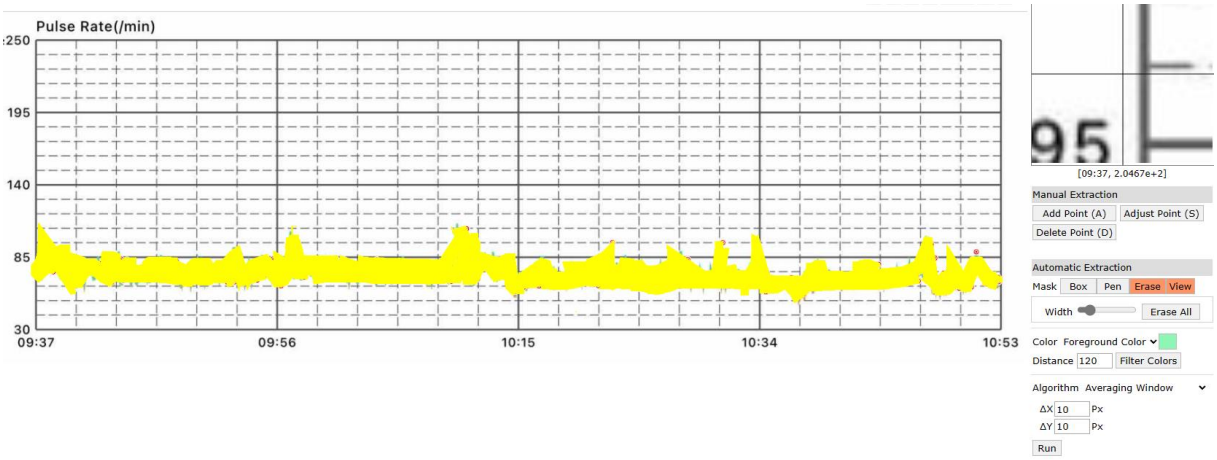
2) Aligner les axes X et Y sur l'image en cliquant sur les points de départ et d'arrivée sur les axes X et Y



3) Entrer des valeurs pour les points sélectionnés

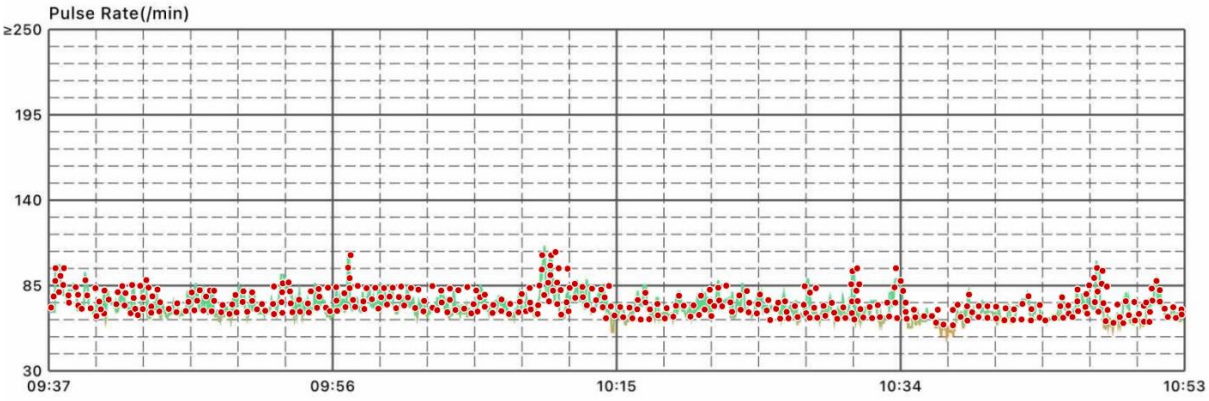


4) Sélectionnez la zone à convertir sur le graphique et algorithme pour convertir. Choisir la couleur de courbe dans le graphique

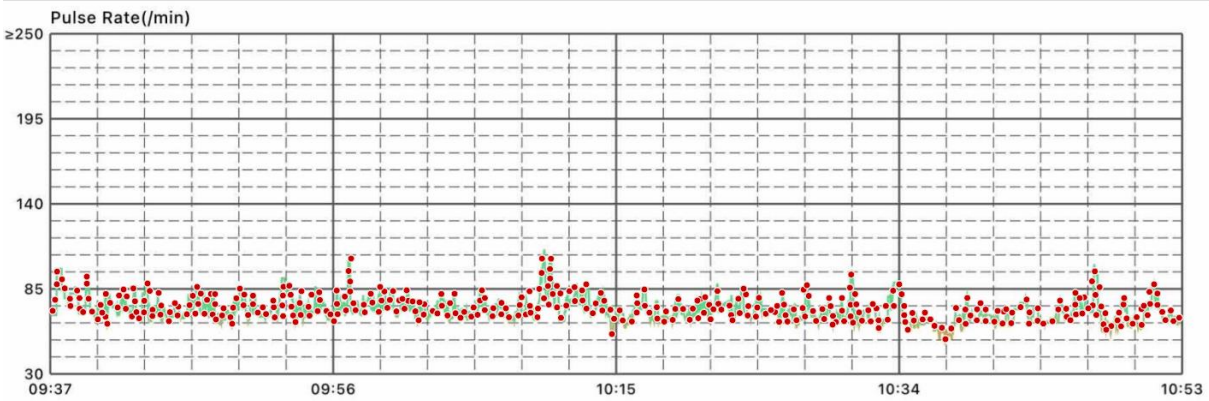


Je choisi Δx , Δy (distance entre 2 points de données selon direction x et y) égales 10 pixels. Si Je diminue Δx , Δy logiciel va détecter plus de points mais aussi plus d'erreurs

5) Exécuter le programme. Le logiciel identifiera les points verts dans la zone sélectionnée avec une fenêtre moyenne = 10. Cependant, comme nous pouvons le voir, ce n'est pas tout à fait exact car il y a des points manquants ou excessifs de la courbe.



6) Supprimez (manuellement) les points inutiles ou mauvais et ajoutez les points nécessaires. Notez que ce dont nous avons besoin est la tendance du changement de fréquence cardiaque (vers le haut ou vers le bas), donc j'ai juste choisi les pics exacts, les valeurs entre les pics pas besoin d'être aussi précis, c'est acceptable à erreur +-1 battre par minute



7) Téléchargez les données au format Excel, chaque point de données dans le graphique ci-dessus i aura des coordonnées (temps, fréquence cardiaque).

09:37	71
09:37	88,1
09:37	96,24
09:38	85,47
09:38	91,26
09:38	78,8
09:38	74,08
09:39	84,03
09:39	79,19
09:39	72,31
09:39	70,13
09:39	88,49
09:40	78,8
09:40	70,49
09:40	65,51
09:40	74,81
09:41	81,87
09:41	67,19
09:41	76,15
09:42	80,87
09:42	73,13

Ainsi, nous avons obtenu des données numériques à partir de l'image. J'ai créé une base de données avec environ 100 fichiers de ce type

2.3.2.2 Formater les données d'oxymetre

Les données obtenues sont déjà sous forme numérique, mais actuellement nous ne pouvons toujours pas travailler avec les données car les chronologies des données sont en rangées verticales. De plus, parfois, ils se mélangent comme indiqué ci-dessous

333	10:49	70,32
334	10:06	84,38
335	09:48	65,46
336	09:47	86,45
337	09:45	64,33
338	10:30	62,44
339	10:28	87,86
340	10:02	65
341	10:33	60,25
342	10:35	59,16
343	10:39	76,33
344	10:41	71,37
345	10:32	75,73
346	10:26	82,87
347	10:25	71,81
348	10:17	85,01
349	10:16	81,14

Pour résoudre ce problème, j'ai écrit du code pour convertir les données dans un format Data Frame avec intervalles de temps comme en-tête des colonnes. Les points de données d'un même intervalle de temps sont regroupés dans la même colonne.

Algorithme : Extraire et Organiser les Données de Séries Temporelles par Intervalle de temps

Entrée : parent_dir (Chemin du répertoire contenant les fichiers CSV)

1. Initialiser un DataFrame vide nommé df_interval.

2. Pour chaque fichier (file) dans le répertoire spécifié (parent_dir) :

Si le fichier a l'extension .csv :

i. Lire le fichier CSV dans un DataFrame nommé df.

ii. Supprimer les lignes avec des valeurs manquantes dans df.

iii. Nettoyer la colonne 'time' en supprimant les espaces en début et en fin, et en extrayant la partie avant le point-virgule.

iv. Convertir la colonne 'value' au format numérique.

v. Convertir la colonne 'time' au format de date et heure, et trier le DataFrame par 'time'.

vi. Initialiser les variables :

- time_interval comme 1 minute

- start_time comme la première valeur de temps dans df

- max_length comme le nombre maximum de points de données dans n'importe quel intervalle d'1 minute

vii. Tant que start_time est inférieur ou égal à la valeur de temps maximale dans df :

A. Calculer end_time en ajoutant le time_interval à start_time.

B. Filtrer les données dans l'intervalle [start_time, end_time) et les stocker dans interval_values.

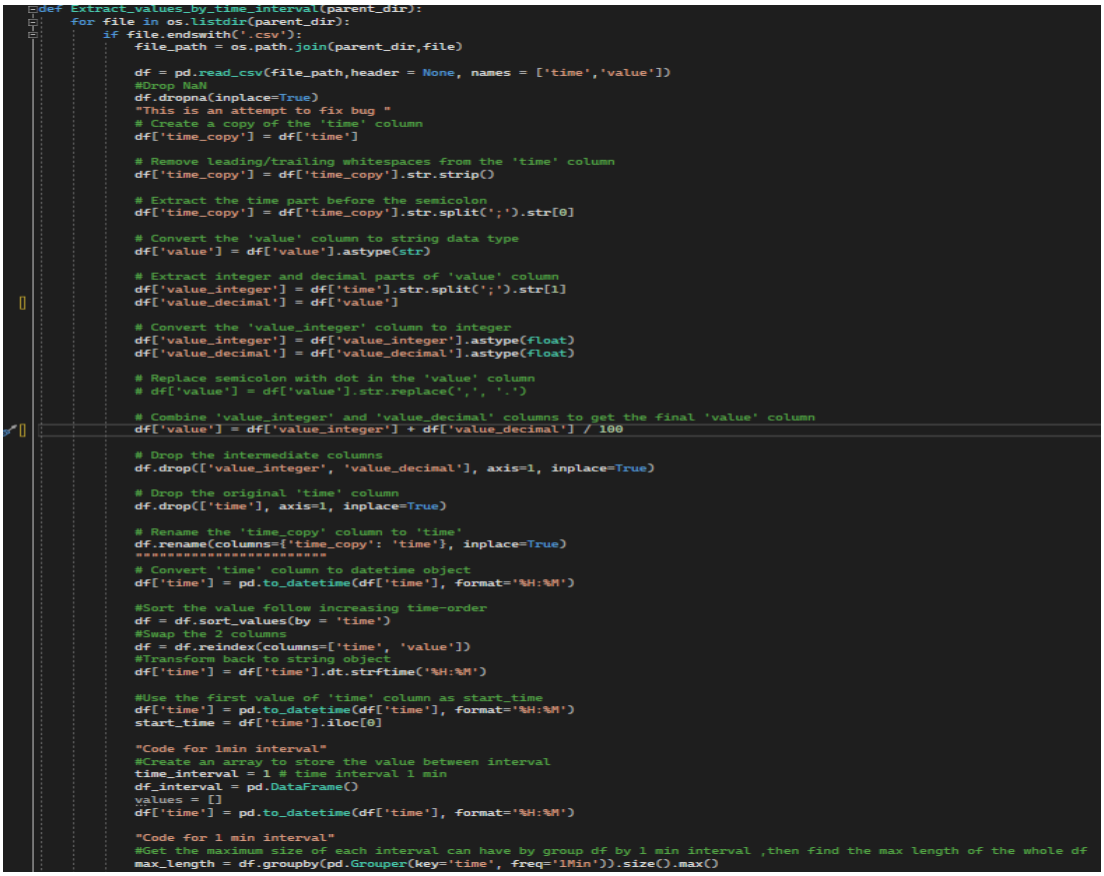
C. Ajouter des valeurs NaN à interval_values pour correspondre à max_length.

D. Créer un en-tête de colonne pour l'intervalle en utilisant start_time et end_time.

E. Ajouter les données 'value' de interval_values à df_interval sous l'en-tête de colonne.

F. Mettre à jour start_time vers end_time.

3. Sortie le DataFrame df_interval, qui contient les données de séries temporelles organisées par intervalles d'1 minute.



```
def Extract_values_by_time_interval(parent_dir):
    for file in os.listdir(parent_dir):
        if file.endswith('.csv'):
            file_path = os.path.join(parent_dir,file)

            df = pd.read_csv(file_path,header = None, names = ['time','value'])
            #Drop NaN
            df.dropna(inplace=True)
            "This is an attempt to fix bug "
            # Create a copy of the 'time' column
            df['time_copy'] = df['time']

            # Remove leading/trailing whitespaces from the 'time' column
            df['time_copy'] = df['time_copy'].str.strip()

            # Extract the time part before the semicolon
            df['time_copy'] = df['time_copy'].str.split(';').str[0]

            # Convert the 'value' column to string data type
            df['value'] = df['value'].astype(str)

            # Extract integer and decimal parts of 'value' column
            df['value_integer'] = df['time'].str.split(';').str[1]
            df['value_decimal'] = df['value']

            # Convert the 'value_integer' column to integer
            df['value_integer'] = df['value_integer'].astype(float)
            df['value_decimal'] = df['value_decimal'].astype(float)

            # Replace semicolon with dot in the 'value' column
            # df['value'] = df['value'].str.replace(',', '.')

            # Combine 'value_integer' and 'value_decimal' columns to get the final 'value' column
            df['value'] = df['value_integer'] + df['value_decimal'] / 100

            # Drop the intermediate columns
            df.drop(['value_integer', 'value_decimal'], axis=1, inplace=True)

            # Drop the original 'time' column
            df.drop(['time'], axis=1, inplace=True)

            # Rename the 'time_copy' column to 'time'
            df.rename(columns={'time_copy': 'time'}, inplace=True)
            """""""""""""""""""""""
            # Convert 'time' column to datetime object
            df['time'] = pd.to_datetime(df['time'], format='%H:%M')

            #Sort the value follow increasing time-order
            df = df.sort_values(by = 'time')
            #Swap the 2 columns
            df = df.reindex(columns=['time', 'value'])
            #Transform back to string object
            df['time'] = df['time'].dt.strftime('%H:%M')

            #Use the first value of 'time' column as start_time
            df['time'] = pd.to_datetime(df['time'], format='%H:%M')
            start_time = df['time'].iloc[0]

            "Code for 1min interval"
            #Create an array to store the value between interval
            time_interval = 1 # time interval 1 min
            df_interval = pd.DataFrame()
            values = []
            df['time'] = pd.to_datetime(df['time'], format='%H:%M')

            "Code for 1 min interval"
            #Get the maximum size of each interval can have by group df by 1 min interval ,then find the max length of the whole df
            max_length = df.groupby(pd.Grouper(key='time', freq='1Min')).size().max()
```

***Sortie** : df_interval (DataFrame contenant les données de séries temporelles organisées par intervalles d'1 minute)*

L'idée de ce code est de convertir les données de la colonne time en datetime à l'aide de la bibliothèque datetime de python. Grâce à cela, nous pouvons trier les valeurs de la colonne 'temps' et comparer les valeurs de temps pour regrouper les valeurs de fréquence cardiaque.

```
while start_time <= df['time'].max():
    #Calculate the start and end time of the interval

    end_time = start_time + timedelta(minutes=time_interval)

    interval_values = df[(df['time'] >= start_time) & (df['time'] < end_time)]

    # Reset index of filtered values
    interval_values = interval_values.reset_index(drop=True)

    # Get the maximum length of the data in the interval
    max_length_interval = interval_values.shape[0]

    # Create column header as interval start time - end time
    column_header = f"{start_time.strftime('%H:%M')} - {end_time.strftime('%H:%M')}"
    #If this is less than max_length, the code appends NaN values to interval_values,if not have this line the df will be truncated
    interval_values = pd.concat([interval_values, pd.DataFrame(index=range(max_length_interval, max_length), columns=interval_values.columns)], ignore_index=True)
    # Append values to df_interval DataFrame with column header
    df_interval.loc[:, column_header] = interval_values['value']
    # df_interval[column_header] = interval_values['value'].dropna()

    # Update start time for next interval as end time + 1 minute
    start_time = end_time
```

Après avoir exécuté le code, nous obtenons un DataFrame comme ci-dessous :

	09:37 - 09:38	09:38 - 09:39	...	10:52 - 10:53	10:53 - 10:54
0	71.00	85.47	...	70.75	67.05
1	88.10	91.26	...	65.51	NaN
2	96.24	78.80	...	71.71	NaN
3	78.09	74.08	...	64.88	NaN
4	NaN	NaN	...	NaN	NaN
5	NaN	NaN	...	NaN	NaN
6	NaN	NaN	...	NaN	NaN
7	NaN	NaN	...	NaN	NaN
8	NaN	NaN	...	NaN	NaN

[9 rows x 77 columns]

Nous pouvons voir que les longueurs des colonnes ne sont pas égales. Cela se produit en raison distribution inégale des points de données sur le graphique à convertir. Pour résoudre ce problème, mon tuteur a suggéré d'utiliser l'interpolation linéaire L'interpolation linéaire est une technique

```
max_length = df_interval.count().max()

longest_columns = []
# Find the columns) with the specified length
for col in df_interval.columns:
    if (df_interval[col]).count() == max_length: #count method exclude NaN value
        longest_columns.append(col)

# print(f'The longest length in the table is: {max_length}')
# print(f'The first column with the length of {max_length} is: {longest_columns}')
#"Choose the first column as reference
ref_column = df_interval[longest_columns[0]]
sorted_indices = np.argsort(ref_column)
# #Perfom interpolation on other column
for col in df_interval.columns:
    column = df_interval[col]
    if column.count() < max_length:
        column = column.interpolate(method = 'linear',axis = 0,inplace = False,limit_direction = 'both')
        #update value
        df_interval[col] = column
```

fondamentale utilisée pour estimer des valeurs dans une plage basée sur des points de données connus. Cela implique de relier deux points de données adjacents par une ligne droite et de prédire des valeurs à des positions entre eux. Cette méthode suppose une relation linéaire entre les points de données, ce qui la rend particulièrement utile pour créer des approximations lorsque les données sont limitées ou inégalement réparties.

Dans mon code, j'ai sélectionné la ligne la plus longue comme ligne de référence, les lignes plus courtes calculeront les approximations basées sur cette ligne. Finalement, nous obtenons des lignes de longueur égale.

9:37 - 09:3	9:38 - 09:3	9:39 - 09:4	9:40 - 09:4	9:41 - 09:4	9:42 - 09:4	9:43 - 09:4	9:44 - 09:4	9:45 - 09:4	9:46 - 09:4	9:47 - 09:4	9:48 - 09:4	9:49 - 09:5	9:50 - 09:5	9:51 - 09:5	9:52 - 09:5	9:53 - 09:5	9:54 - 09:5
71	85,47	93,03	78,8	62,64	80,26	85,47	88,64	75,94	73,69	86,45	68,61	72,66	85,37	68,98	77,65	63,84	67,97
88,1	91,26	88,49	70,49	67,19	72,32	77,17	82,57	70,15	68,83	69,1	81,92	73,04	68,37	73,31	66,86	86,64	73,64
96,24	78,8	72,31	65,51	76,15	67,38	70,16	74,42	64,33	74,69	82,15	78,08	66,92	74,63	69,9	73,38	74,56	68,45
78,09	74,08	79,19	74,81	81,87	84,76	66,16	68,52	64,33	68,03	75,46	65,46	68,01	81,51	75,32	73,38	67,69	76,16
78,09	74,08	84,03	69,82	81,87	80,87	77,42	67,65	64,33	68,03	69,21	81,82	79,1	81,51	80,78	73,38	68,07	83,74
78,09	74,08	70,13	69,82	81,87	73,13	70,05	71,7	64,33	68,03	80,71	75,31	62,7	81,51	80,78	73,38	81,43	83,74
78,09	74,08	70,13	69,82	81,87	73,13	70,05	83,87	64,33	68,03	80,71	75,31	62,7	81,51	80,78	73,38	87,24	83,74
78,09	74,08	70,13	69,82	81,87	73,13	70,05	83,87	64,33	68,03	80,71	75,31	62,7	81,51	80,78	73,38	73,74	83,74
78,09	74,08	70,13	69,82	81,87	73,13	70,05	83,87	64,33	68,03	80,71	75,31	62,7	81,51	80,78	73,38	80,92	83,74

Fig.12 : Les données d'image ont été converties en excel

2.3.2.3 Extraire les visages de la vidéo

Mon prochain travail consiste à convertir des données vidéo en données d'image. L'objectif est que dans chaque vidéo d'une leçon, je choisisse 5 photos de 5 personnes comme images de référence. Ensuite, je lirai la vidéo, pour chaque frame correspondant, j'identifierai chaque visage dans ce frame, je compare chaque visage dans ce frame avec l'image de référence pour trouver l'image la plus similaire en calculant la distance euclidienne entre 2 images

Mon tuteur me demande d'enregistrer des images à vitesse fixe de 2 images/seconde, soit 120 images/minute. Cela signifie que nous devons toujours capturer le visage d'une personne à tout moment de la vidéo. Au début, j'ai utilisé le classificateur Haar Cascade pour identifier les visages humains dans chaque image. Cependant, comme je l'ai dit au début, HaarCascadeClassifier s'appuie fortement sur les modèles de pixels pour l'identification, donc cela ne fonctionne pas vraiment lorsque la personne tournée sur les côtés ou a un teint foncé.

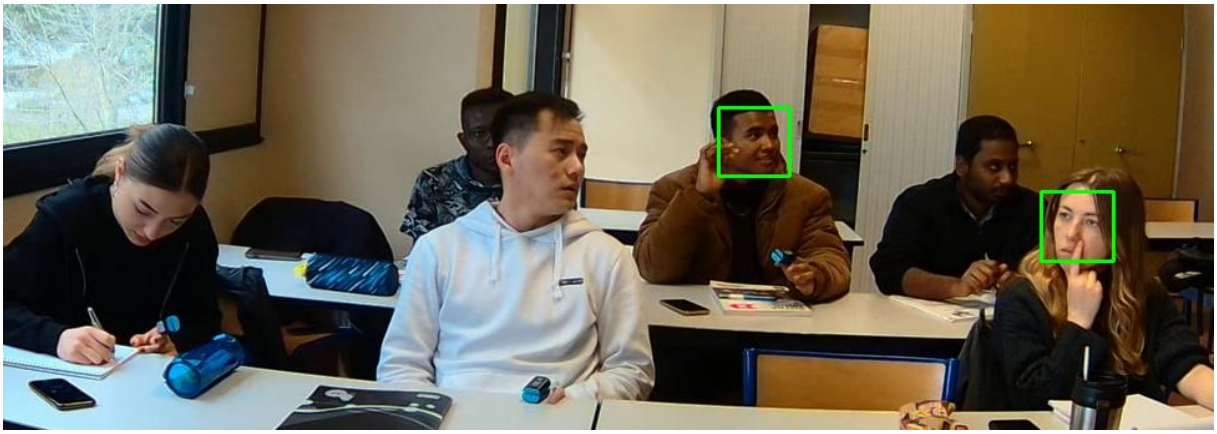


Fig 13 : Exemple de detection de faces en video en utilisant HaarCascadeClassifier

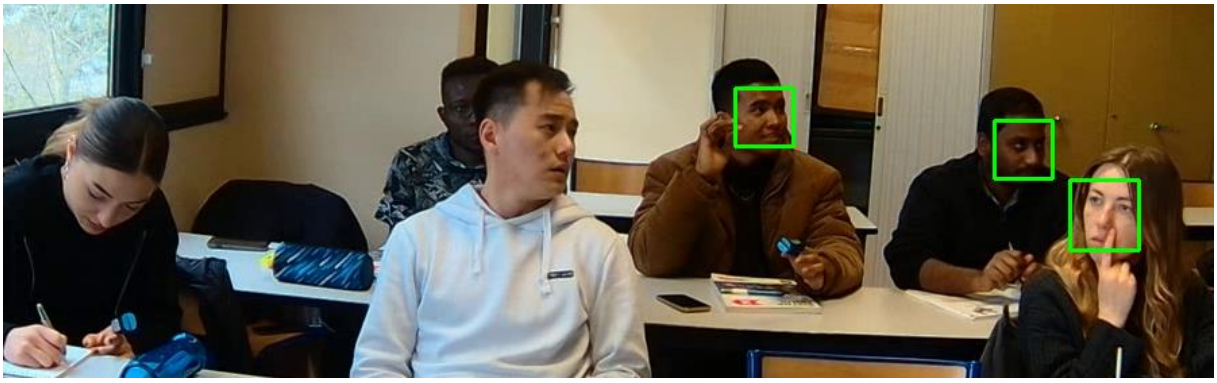


Fig.14 : Exemple de detection de faces en video en utilisant VGGFaces2

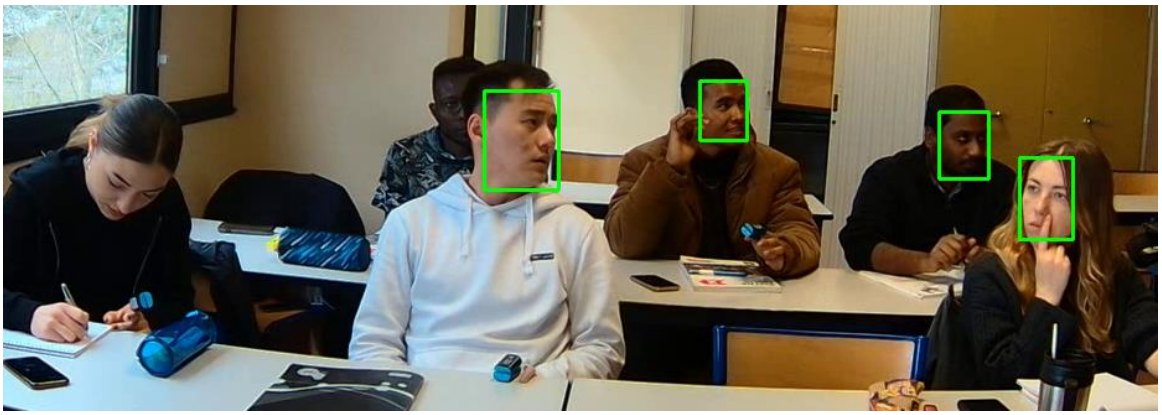


Fig.15 : Exemple de detection de faces en video en utilisant MTCNN

Comme nous pouvons le voir, MTCNN a les meilleures performances parmi les 3 détecteurs de visage typiques donc dans ce projet j'ai utilisé MTCNN. Pour comparer des visages detectes avec des images references j'utilisé face-recognition. La bibliothèque face-recognition est un package Python populaire qui fournit une interface de haut niveau pour la détection des visages, la reconnaissance des visages et la manipulation des caractéristiques faciales. Il utilise des modèles d'apprentissage en profondeur pré-formés pour effectuer des tâches liées à l'analyse des visages. La bibliothèque est construite au-dessus de la bibliothèque Dlib et utilise ses puissantes capacités de reconnaissance faciale.

Algorithme : Détection et Comparaison de Visages en utilisant face-recognition et MTCNN

i. Initialisation :

- *Importer les bibliothèques requises : cv2, face_recognition, numpy, os.*
- *Charger la vidéo en utilisant cv2.VideoCapture.*
- *Calculer la vitesse de lire frame par seconde (fps) de la vidéo.*
- *Définir l'intervalle de trame (par exemple, toutes les 0,5 secondes).*
- *Définir l'heure de début souhaitée en heures, minutes et secondes.*

***ii. Charger et Prétraiter les Images de Référence** :*

- *Initialiser une liste vide reference_encodings.*
- *Pour chaque chemin d'image de référence :*
  - *Charger l'image en utilisant cv2.imread et convertir en RVB.*
  - *Détecter les emplacements des visages en utilisant face_recognition. face_locations.*
  - *Si des visages sont détectés :*
    - *Calculer l'encodage du visage en utilisant face_recognition. face_encodings.*
    - *Ajouter l'encodage du visage à reference_encodings.*
  - *Afficher un message si aucun visage n'est détecté.*

***iii. Traiter les Images de la Vidéo et Détecter les Visages** :*

- *Initialiser le compteur de trames, les secondes écoulées et les images enregistrées.*

- *Boucler à travers chaque trame de la vidéo en utilisant* cap. isOpened() *:*

  - *Lire la trame en utilisant cap. read.*

  - *Convertir la trame en format RVB.*

  - *Si le compteur de trames < trame de début, incrémenter le compteur de trames et continuer.*

  - *Si le compteur de trames % intervalle de trame == 0 :*

    - *Détecter les visages dans la trame en utilisant MTCNN.*

    - *Pour chaque résultat de visage détecté :*

      - *Extraire la région du visage et redimensionner.*

      - *Calculer l'encodage du visage en utilisant face_recognition. face_encodings.*

      - *Comparer l'encodage du visage avec les encodages de référence.*

      - *Si aucune correspondance exacte :*

        - *Calculer les distances cosinus avec les encodages de référence.*

        - *Choisir la correspondance la plus proche en dessous d'un seuil (0.7).*

      - *Si une correspondance est trouvée :*

        - *Redimensionner l'image du visage.*

        - *Générer un nom de fichier basé sur l'horodatage, le numéro de trame, les minutes et les secondes.*

        - *Enregistrer l'image du visage dans le répertoire de sortie en utilisant cv2.imwrite.*

        - *Dessiner un rectangle autour du visage détecté dans la trame en utilisant cv2.rectangle.*

        - *Afficher l'horodatage et le nombre d'images enregistrées.*

iv. Libérer les Ressources et Fermer les Fenêtres :

- *Libérer la capture vidéo en utilisant cap.release().*

- *Fermer les fenêtres cv2 ouvertes en utilisant cv2.destroyAllWindows().*

v. Fin de l'Algorithme.

```python
from keras_facenet import FaceNet
import face_recognition
from mtcnn.mtcnn import MTCNN
import os
from numba import  cuda
devices = cuda.list_devices()
# Print the device information
for i, device in enumerate(devices):
    print(f"Device {i}: {device.name}")
cuda.select_device(0)

# Print the selected device information
device = cuda.get_current_device()
print(f"Selected GPU: {device.name}")
# Load the FaceNet model
embedder = FaceNet()

# Initialize the MTCNN model for face detection
detector = MTCNN()

# Open the video file
cap = cv2.VideoCapture(r'E:\Projet 2023\Data\Video\23-03-2023-session-2.mp4')
output_dir = r'E:\Projet 2023\Data\Faces_data\Micheal\23-03-2023-FaceNet'

# Get the frames per second (fps) of the video
fps = cap.get(cv2.CAP_PROP_FPS)
frame_interval = int(fps / 2)

# Print the fps
print(f"Frames per second (fps): {fps}")

# Skip frames until the desired starting point
start_time = 0
start_frame = int(start_time * fps)
for _ in range(start_frame):
    cap.read()

frame_counter = 0
elapsed_seconds = start_time
frames_saved = 0

# Load and preprocess the reference images
reference_paths = [
    r"E:\Projet 2023\Data\Faces_data\Reference_Images\Micheal1.jpg",
    r"E:\Projet 2023\Data\Faces_data\Reference_Images\Micheal2.jpg",
    r"E:\Projet 2023\Data\Faces_data\Reference_Images\Micheal3.jpg"
]

reference_encodings = []

for path in reference_paths:
    image = cv2.cvtColor(cv2.imread(path), cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(image)

    if len(face_locations) > 0:
        encoding = face_recognition.face_encodings(image, face_locations)[0]
        reference_encodings.append(encoding)
    else:
        print(f"No face detected in the reference image: {path}")
```

```python
while cap.isOpened():
    ret, frame = cap.read()

    if not ret:
        break

    if frame_counter < start_frame:
        frame_counter += 1
        continue

    if frame_counter % frame_interval == 0:
        # Convert the frame to RGB format (required by MTCNN)
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Detect faces using MTCNN
        results = detector.detect_faces(rgb_frame)
        print("There are ",len(results),' faces detected')
        if len(results) > 0:
            for result in results:
                x, y, w, h = result['box']
                face = cv2.resize(rgb_frame[y:y+h, x:x+w], (160, 160))

                # Encode the detected face
                detected_face_encodings = face_recognition.face_encodings(face)
                if len(detected_face_encodings) > 0:
                    detected_face_encoding = detected_face_encodings[0]
                    # Compare face encodings with reference encodings
                    matches = face_recognition.compare_faces(reference_encodings, detected_face_encoding, tolerance=0.7)
                    print(matches)
                    if not any(matches):
                        # Calculate cosine distance with reference encodings and choose the closest match
                        distances = [np.linalg.norm(detected_face_encoding - ref_encoding) for ref_encoding in reference_encodings]
                        print(distances)
                        closest_match_index = np.argmin(distances)
                        if distances[closest_match_index] < 0.7:
                            matches[closest_match_index] = True
                    if any(matches):
                        print("There is 1 match")
                        face_image = cv2.resize(face, (128, 128))
                        frame_number = frame_counter // frame_interval + 1
                        minutes = int(elapsed_seconds // 60)
                        seconds = int(elapsed_seconds % 60)
                        filename = f"min_{minutes:02d}_second_{seconds:02d}_frame_{frame_number:04d}.jpg"
                        cv2.imwrite(os.path.join(output_dir, filename), face_image)
                        frames_saved += 1
                        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                        print(f'{minutes:02d}:{seconds:02d} have passed, {frames_saved} images saved')


    frame_counter += 1
    elapsed_seconds = frame_counter / fps

    # cv2.imshow("Video", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release video capture and close windows
cap.release()
```

Voici les resultats du code en ci-dessus :

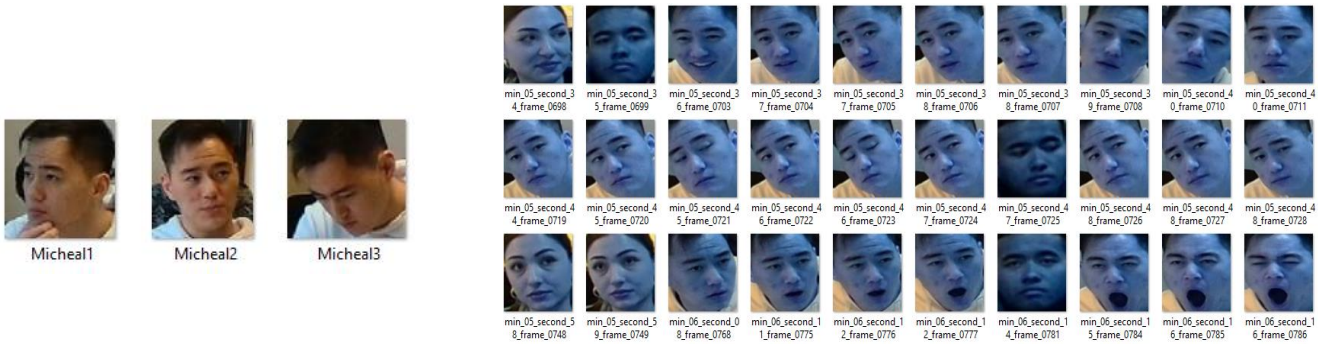


Fig.16 : 3 photos pour référence (à gauche) et les et les faces obtenues par l'algorithme (à droite)

On voit que la plupart des photos de visage extraites appartiennent à la même personne, cependant il y a encore quelques images qui sont confondues avec d'autres personnes, la raison est que parfois la personne ne regarde pas directement sur le camera ou la tête baissée. Pour résoudre ce problème, J'ai créé une liste d'images de référence de temps en temps dans la vidéo. Toutes les 3 minutes, je capture une photo de cette personne pour être référence. Chaque photo de visage obtenue en 3 minutes sera ensuite comparée à cette photo de référence. La raison en est que, selon mon observation, cette personne change de position toutes les 3 minutes, donc chaque image de ces 3 minutes sera essentiellement la même, ce qui augmente la précision.

Avec la même idée que ci-dessus, j'ai écrit un code pour générer une liste de références basée sur les horodatages de la vidéo. Certaines références incorrectes ont été corrigées manuellement

Algorithme : Extraction de Visages à partir de Vidéos avec référence de temps correspondant

i. Charger et prétraiter les images de référence :

- *Pour chaque fichier dans le répertoire de référence :*
  - *Lire l'image depuis le fichier*
  - *Convertir l'image au format RGB*
  - *Normaliser et remodeler l'image pour le modèle facenet*
  - *Calculer l'incorporation à l'aide du modèle facenet*
  - *Extraire les informations d'intervalle du nom de fichier*
  - *Ajouter l'intervalle et le descripteur à la liste reference_info*

ii. Initialiser les variables frame_counter, images_saved et saved_image_info

iii. Traiter les images vidéo :

- Tant qu'il y a des images dans la vidéo :

- Lire l'image suivante

- Calculer l'heure actuelle (minute et seconde)

- Calculer l'intervalle actuel en fonction de la minute actuelle

- Si le frame_counter est à un intervalle approprié et que current_interval n'est pas dans reference_info ou que current_interval est égal à 0 :

- Convertir l'image en RVB

- Détecter les visages dans l'image à l'aide du détecteur

- Si des visages sont détectés :

- Redimensionner et prétraiter chaque visage détecté

- Calculer le descripteur de visage pour chaque visage

- Obtenir les descripteurs de référence correspondants pour current_interval

- Calculer les distances entre les visages détectés et les descripteurs de référence

- Trouver le visage le plus adapté en fonction de la distance minimale et < 0.7

- S'il y a une correspondance valide :

- Enregistrer l'image du visage correspondant dans le répertoire de sortie

- Mettre à jour images_saved et saved_image_info

- Ajouter les nouvelles informations de référence à reference_info

- Incrémenter le frame_counter

iv. Libérer la capture vidéo et fermer les fenêtres

v. Afficher "Extraction terminée."

Pour maximiser la vitesse de traitement, le code sera exécuté sur le GPU pour ordinateur portable NVIDIA GeForce RTX 3050. Au lieu de devoir utiliser facenet pour comparer à chaque fois ce qui prend beaucoup de temps, maintenant on peut simplement calculer la distance euclidienne entre les deux encodages de visage et référence. Cela augmente considérablement la vitesse de traitement tout en gardant une précision assez élevée

```
for filename in os.listdir(reference_dir):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        image_path = os.path.join(reference_dir, filename)
        reference_image = cv2.imread(image_path)
        reference_images.append(reference_image)

        reference_face = cv2.cvtColor(reference_image, cv2.COLOR_BGR2RGB)
        reference_face = torch.tensor(reference_face, dtype=torch.float32).permute(2, 0, 1) / 255
        reference_face = reference_face.unsqueeze(0)
        reference_embedding = facenet(reference_face).detach().numpy()
        current_interval = int(filename.split('_')[1].split('.')[0])
        # print('current_interval',current_interval)
        reference_info.append({'interval': current_interval, 'descriptor': reference_embedding})
```

```python
if frame_counter % (fps / 2) == 0 and current_interval not in [info['interval'] for info in reference_info] or current_interval == 0:
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = detector.detect_faces(rgb_frame)

    if len(results) > 0:
        face_images = []
        for result in results:
            x, y, w, h = result['box']
            face = cv2.resize(rgb_frame[y:y+h, x:x+w], (160, 160))
            face_images.append(face)

        face_descriptors = []
        for face_image in face_images:
            face_image = torch.tensor(face_image, dtype=torch.float32).permute(2, 0, 1) / 255
            face_image = face_image.unsqueeze(0)
            face_descriptor = facenet(face_image).detach()
            face_descriptors.append(face_descriptor)
        best_match_distance = 0
        # Compare each face descriptor to the single reference descriptor for the interval
        matching_info_interval = [info for info in reference_info if info['interval'] == current_interval]
        distances = [np.linalg.norm(face_descriptor - info['descriptor']) for info in matching_info_interval]
        for i,distance in  enumerate(distances):
            if distance > best_match_distance and distance < 0.7:
                # Find the index of the minimum distance, which corresponds to the best match4
                best_match_distance = distance
                best_match_index = i
            print("best match index: ",best_match_index)
            if  best_match_index is not None and best_match_index < len(face_images):

                # Save matching face as a reference
                image_info = f"face_{{images_saved:04d}.jpg"
                save_path = os.path.join(output_dir, image_info)
                resized_face_image = cv2.resize(face_images[best_match_index], (128, 128))
                cv2.imwrite(save_path, cv2.cvtColor(resized_face_image, cv2.COLOR_RGB2BGR))
                images_saved += 1
                saved_image_info.append(image_info)

                # Add the new reference info
                reference_info.append({'interval': current_interval, 'descriptor': face_descriptor})

frame_counter += 1
```

2.4Analyse des données

2.4.1 Analyse donnée d'oxymètre

2.4.1.1 Similarité

Mon travail après avoir une base de données complet est d'analyser la similarité des données dans chaque fichier. En science des données, la mesure de similarité est un moyen de mesurer la façon dont les échantillons de données sont liés ou proches les uns les autres. D'autre part, la mesure de dissimilarité est de dire à quel point les objets de données sont distincts.

La mesure de similarité est généralement exprimée sous la forme d'une valeur numérique .Une baisse ou une augmentation de la similarité entre deux colonnes de donnés adjacentes signifierait un changement dans les modèles sous-jacents ou un changement dans le processus de génération de données, ici est la variation de la fréquence cardiaque. De grandes variations de la fréquence cardiaque peuvent être le résultat d'un stimulus, entraînant souvent des changements émotionnels.

9:37 - 09:3	9:38 - 09:3	9:39 - 09:4	9:40 - 09:4	9:41 - 09:4	9:42 - 09:4	9:43 - 09:4	9:44 - 09:4	9:45 - 09:4	9:46 - 09:4	9
71	85,47	93,03	78,8	62,64	80,26	85,47	88,64	75,94	73,69	
88,1	91,26	88,49	70,49	67,19	72,32	77,17	82,57	70,15	68,83	
96,24	78,8	72,31	65,51	76,15	67,38	70,16	74,42	64,33	74,69	
78,09	74,08	79,19	74,81	81,87	84,76	66,16	68,52	64,33	68,03	
78,09	74,08	84,03	69,82	81,87	80,87	77,42	67,65	64,33	68,03	
78,09	74,08	70,13	69,82	81,87	73,13	70,05	71,7	64,33	68,03	
78,09	74,08	70,13	69,82	81,87	73,13	70,05	83,87	64,33	68,03	
78,09	74,08	70,13	69,82	81,87	73,13	70,05	83,87	64,33	68,03	
78,09	74,08	70,13	69,82	81,87	73,13	70,05	83,87	64,33	68,03	

Dans ce projet, nous calculerons des valeurs de fréquence cardiaque entre 2 minutes consécutives du début à la fin du fichier. Il existe environ 17 types de distance différents, mais nous n'utilisons que les deux plus courants, Eucliden et Pearsonr

La fonction de distance la plus couramment utilisée pour les attributs ou entités numériques est la distance euclidienne qui est définie dans la formule suivante :

$$d(P,Q) = ||P-Q||_0 = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$
$$= \sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \ldots + (p_n-q_n)^2}$$

where:

$P = (p_1, p_2, \ldots, p_n)$, and $Q = (q_1, q_2, \ldots, q_n)$

Fig.17 : Formule pour calculer la distance Eucliden entre 2 vecteur

La distance euclidienne est très populaire en raison de sa simplicité. Cependant, il est très sensible à l'échelle de la variable et à la sensibilité aux valeurs aberrantes

Pearson correlation distance peut résoudre ce problème. Le r de Pearson est invariant à l'échelle, ce qui signifie qu'il n'est pas affecté par les changements d'échelle des données et est aussi moins sensible aux valeurs aberrantes que certaines autres mesures de corrélation.

La formule de Corelation distance ou Pearson distance :

$$Correlation\_Distance = 1 - Correlation\_Similarity$$
$$= 1 - \frac{Covariance(P,Q)}{\sqrt{Variance(P)} \cdot \sqrt{Variance(Q)}}$$
$$= 1 - \frac{\sum_{i=1}^{n}(p_{ij} - (1/n)\cdot\sum_{j=1}^{n} p_{ij}) \cdot (q_{ij} - (1/n)\cdot\sum_{j=1}^{n} q_{ij})}{\sqrt{\sum_{i=1}^{n}(p_{ij} - (1/n)\cdot\sum_{j=1}^{n} p_{ij})^2} \cdot \sqrt{\sum_{i=1}^{n}(q_{ij} - (1/n)\cdot\sum_{j=1}^{n} q_{ij})^2}}$$

Fig.18 : Formule pour calculer la distance Pearsonr entre 2 vecteur

Application des formules dans le code

```
def Calculate_similarities_Euclidean(parent_dir):
    # Initialize an array to store similarities
    similarities_euclidean = []
    for filename in os.listdir(parent_dir):
        if not filename.startswith("~$") and filename.endswith(".xlsx") and filename.startswith('output'): #Use startswith to avoid temporary file
            file_path = os.path.join(parent_dir,filename)

            #Read excel file
            df = pd.read_excel(file_path,engine='openpyxl',sheet_name = 'Sheet1')

            # Initialize an array to store similarities

            #Name of the person
            name =  os.path.basename(parent_dir)

            #Loop through columns
            for i in range(0,df.shape[1]-1):
                j = i + 1
                col1 = df.iloc[1:,i]
                col2 = df.iloc[1:,j]
                similarity = (sum(pow(a-b,2) for a, b in zip(col1, col2)))**0.5
                similarities_euclidean.append(similarity)

    return similarities_euclidean
```

```
def Calculate_similarities_Pearsonr(parent_dir):

    similarities_pearsonr = []
    #AFind the excel file path
    for filename in os.listdir(parent_dir):
        if not filename.startswith("~$") and filename.endswith(".xlsx") and filename.startswith('output'): #Use startswith to avoid temporary file
            file_path = os.path.join(parent_dir,filename)

            #Read excel file
            df = pd.read_excel(file_path,engine='openpyxl',sheet_name = 'Sheet1')

            #Name of the person
            name =  os.path.basename(parent_dir)

            #Loop through columns
            for i in range(0,df.shape[1]-1):
                j = i + 1
                col1 = df.iloc[1:,i]
                col2 = df.iloc[1:,j]
                if len(col1) >= 2 and len(col2) >= 2 and len(col1) == len(col2):
                    similarity, _ = pearsonr(col1, col2)
                    similarities_pearsonr.append(similarity)

    return similarities_pearsonr
```

J'ai également écrit une fonction qui permet d'écrire 2 similitudes sur une nouvelle feuille dans le fichier de données d'origine.

```python
def Write_similarities_to_sheet(parent_dir,similarities_euclidean,similarities_pearsonr):
        # Write the similarity results to Excel
        # Write to output.xlsx
        workbook_path  = os.path.join(parent_dir,'output.xlsx')
        if os.path.exists(workbook_path):

            workbook = openpyxl.load_workbook(workbook_path)
            #Ceate a new sheet in excel
            new_sheet = workbook.create_sheet('Similarities')
            # # Select the sheet you want to write to
            worksheet = new_sheet

            # # # Get the last row of the table
            last_row = 10

            # # # Skip one row from the last row of the table
            next_row = last_row + 2

            # # # Write the label 'Pearsonr Method' on the left of the next row
            label_cell_euclidean = worksheet.cell(row=next_row, column=1)
            label_cell_euclidean.value = 'Euclidean Method'
            label_cell_euclidean.font = Font(bold = True)

            # # # Write the values of the array on the row below the table
            for i, value in enumerate(similarities_euclidean):
                worksheet.cell(row=next_row + 1, column=i+1).value = value

            next_row = next_row + 2

            # # # Write the label 'Pearsonr Method' on the left of the next row
            label_cell_pearson = worksheet.cell(row=next_row, column=1)
            label_cell_pearson.value = 'Pearsonr Method'
            label_cell_pearson.font = Font(bold = True)

            # # # Write the values of the array on the row below
            for i, value in enumerate(similarities_pearsonr):
                worksheet.cell(row=next_row + 1, column=i+1).value = value

            # # # Save the changes to the Excel file
            workbook.save(workbook_path)

        workbook_path_1  = os.path.join(parent_dir,'output1.xlsx')

        if os.path.exists(workbook_path_1):
            print(1)
            workbook = openpyxl.load_workbook(workbook_path_1)

            #Ceate a new sheet in excel
            new_sheet = workbook.create_sheet('Similarities')

            # # Select the sheet you want to write to
            worksheet = new_sheet

            # # # Get the last row of the table
            last_row = len(df)

            # # # Skip one row from the last row of the table
            next_row = last_row + 2

            # # # Write the label 'Pearsonr Method' on the left of the next row
            label_cell_euclidean = worksheet.cell(row=next_row, column=1)
            label_cell_euclidean.value = 'Euclidean Method'
            label_cell_euclidean.font = Font(bold = True)

            # # # Write the values of the array on the row below the table
```

Euclidean Method									
20,26376	15,40512	24,32556	29,99926	20,44828	20,66541	26,9078	38,52784	13,82794	27,48548
Pearsonr Method									
0,618935	0,660073	0,31617	0,197208	0,383029	-0,0791	0,014332	0,307688	-0,02297	0,286931

Sheet1 | Similarities | +

Ci-dessous sont graphique de changement des similarité euclidienne et Pearsonr selon les donnés en haut à droite :

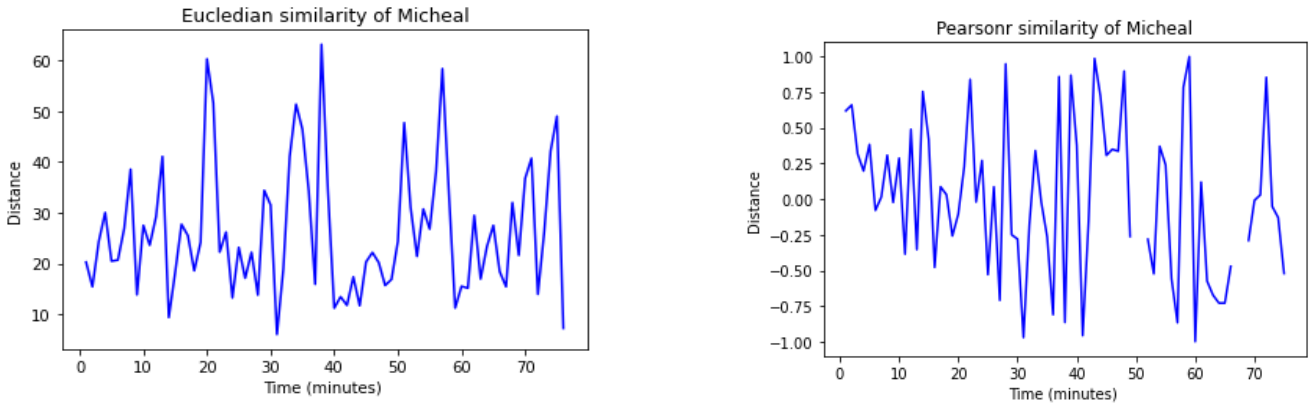


Fig 19 : Similarité Euclidiennes (à gauche) et similarités Pearsonr (à droite) graphique d'une session d'étude

Euclidien similarité :

On observe que dans le graphe euclidien, il y a 3 pics où la distance change significativement : environ minute 20 , minute 37 et 57 minutes. On peut dire qu'il y a un changement brutal de fréquence cardiaque sur les intervalles de cette durée et peut être le résultat de changements émotionnels.

Cependant, il peut aussi être causé par un changement brusque de position ou par une commotion cérébrale à l'oxymètre.

Pearsonr similarité :

Dans pearsonr similarité ,1 sens une corrélation positive parfaite et -1 sens une corrélation négative parfaite . Si le r de Pearson est proche de 1, cela indique une forte relation linéaire positive entre les variables. Sur le graphique, on voit que le graphique est dans la partie positive pendant environ 5 minutes avant de redevenir négatif et de nouveau positif, ce qui s'avère montre qu'en 5 minutes, cela indique une forte relation linéaire positive entre les variables

Nous pouvons également utiliser la similarité pour suivre la similarité du modèle de fréquence cardiaque de chaque personne au cours d'un semestre. Ci-dessous se trouve le diagramme euclidien similaire de Michael et Elena. Michael est une personne très active et bavarde en classe, tandis qu'Elena est au contraire très calme et tranquille. Nous pouvons voir une nette différence entre les 2 modèles de graphique

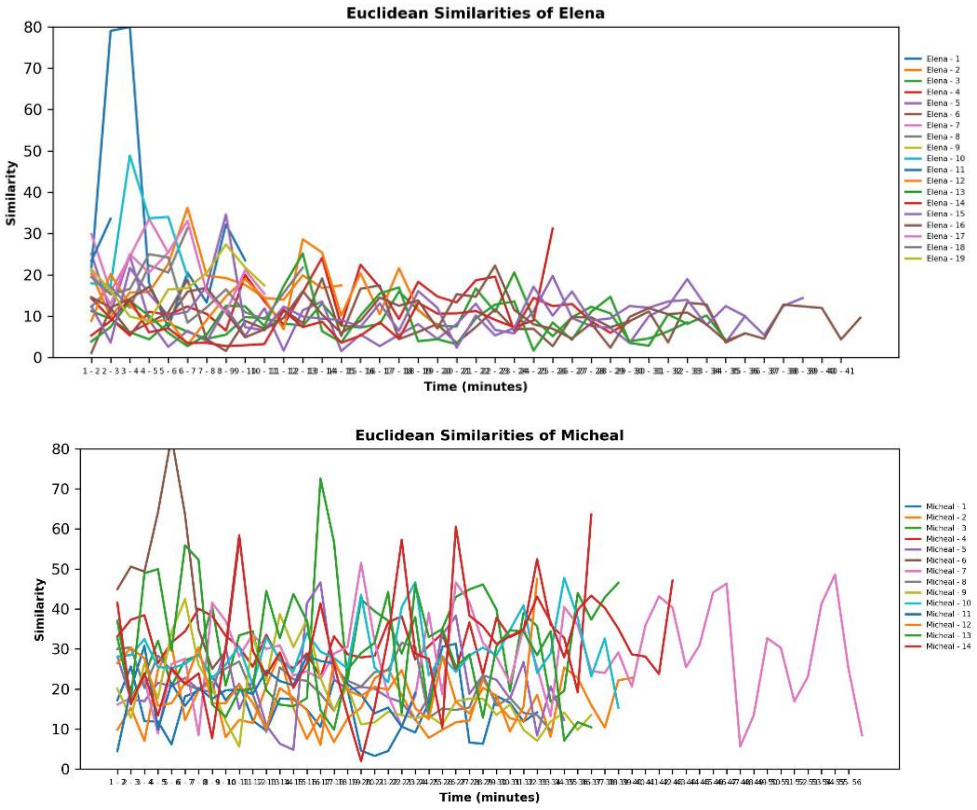


Fig 20 : Graphique de similarité euclidienne de toutes les leçons par Michael (haut) et Elena (bas)

2.4.1.2 Analyse basée sur la méthode de l'état de repos

Une autre méthode pour analyser les données de l'oxymètre consiste à utiliser la fréquence cardiaque à un état stable. Une autre méthode d'analyse des données de l'oxymètre consiste à utiliser la fréquence cardiaque dans un état stationnaire, inactif ou agité. Selon le rapport scientifique « Estimation of heart rate and respiratory rate from the seismocardiogram

under resting state » , La fréquence cardiaque au repos d'un jeune adulte en bonne santé âgé de 20 à 30 ans peut varier de **57 à 80 bpm** (voir figure ci-dessous).

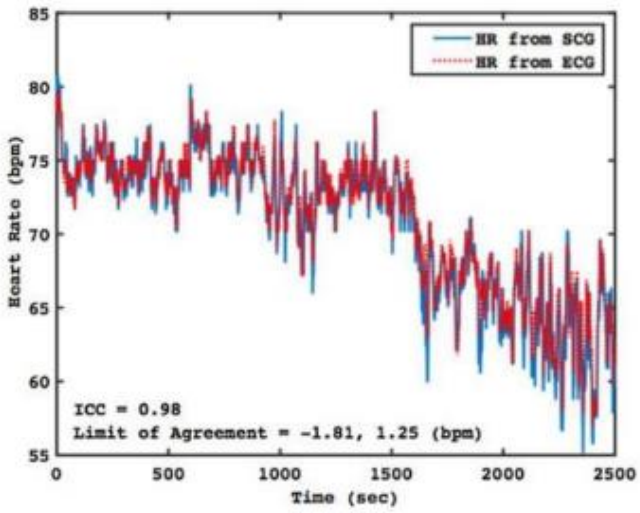


Fig 21 : Graphique de fréquence cardiaque statique au fil du temps, affiché dans le rapport [1]

Ainsi, nous pouvons dire que si la fréquence cardiaque est en dehors de la plage 57-80, la personne peut ne pas être dans un état statique. Bien sûr, nous ne pouvons pas garantir un changement émotionnel, mais nous pouvons deviner à ces moments-là que certaines activités physiques (se lever, rire, parler...) qui peut changent les émotions. J'ai écrit du code pour illustrer cela, notez que les points sont la valeur moyenne de chaque ligne (minutes) et le nombre ci-dessus est le nombre de minutes.

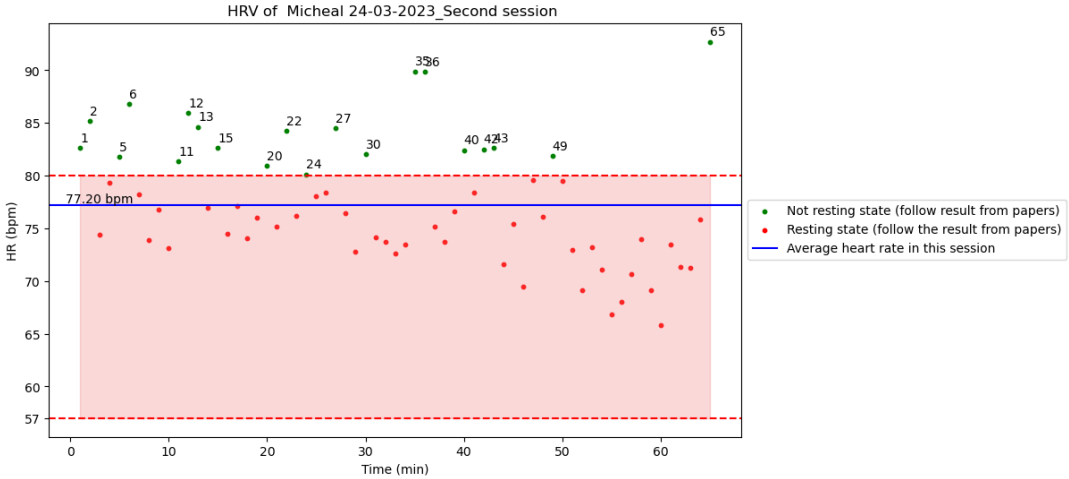


Fig. 22 : Illustrer les minutes de fréquence cardiaque moyenne en dehors de l'état de repos (vert) et à l'intérieur de l'état de repos (rouge)

Nous voyons que la fréquence cardiaque moyenne dans la figure ci-dessus est de 77 et est proche de 80. Dans le groupe d'étude, il y avait un éleve avec des fréquences cardiaques très élevées (Micheal) et un très basses (Elena), donc pour mieux s'adapter à la situation, mon tuteur m'a suggéré d'analyser le pourcentage de points de données qui a valeur plus grande que la fréquence cardiaque moyenne dans toutes les sessions d'études.

Voici les résultats :

Bayu	Elena	Micheal	Tamily Falcao	Yeliz
45.02 %	46.29%	**45.5%**	40.63%	46.16 %

2.4.1.3 Prédiction des emotion a base de données de oxymetre en utilisant KMeans clustering

Après analyse des données, le travail final consiste encore à utiliser le clustering KMeans pour prédire les émotions. La prédiction ici ne dit pas exactement les sentiments de la personne tels que : heureux, triste, ennuyé, ... Nous ne pouvons dire qu'à cette minute, les sentiments de cette personne appartiennent à un certain groupe., si vous voulez savoir ce que ce groupe d'émotions c'est-à-dire qu'il est nécessaire d'avoir des entretiens en face-à-face pour le déterminer. Dans ce rapport, je ne présente que la méthode pour les regrouper.

Ce processus se compose de 6 étapes principales

Étape 1 : Pour chaque personne, fusionnez tous les fichiers de données de l'oxymètre pour fournir un grand fichier. La dernière ligne de ce fichier sera la ligne suivante du fichier suivant.

```
for file_name in file_list:
    file_path = os.path.join(folder_path, file_name)
    df = pd.read_excel(file_path,sheet_name='Sheet1')
    headers = df.columns  # Skip the first column (timestamps)
    all_headers.extend(headers)


# Find non-existing headers and add them to a separate header group
# Remove duplicates and sort the headers
all_headers = sorted(set(all_headers))

max_length = 0
#Create DataFrame with header is complete all_header
merged_df = pd.DataFrame(index = range(20), columns=all_headers,dtype = float) #have to specify the datatype

total_length = 0
merged_data = {header: [] for header in all_headers}

for file_name in file_list:
    file_path = os.path.join(folder_path, file_name)
    df = pd.read_excel(file_path,sheet_name = 'Sheet1')
    file_length = sum(len(df[column]) for column in df.columns)
    total_length += file_length
    for column in df.columns:
        timestamp = column
        # Check if the column header is a timestamp
        if " - " in column:
            column_data = pd.to_numeric(df[column].reset_index(drop=True), errors='coerce')
            # Check if the column header is already present in the merged DataFrame
            # if timestamp in merged_df.columns:
            if timestamp in all_headers:
                # Concatenate the column data with the existing values in the merged DataFrame
                # column_data = pd.Series(column_data)
                # concatenated_data = pd.concat([merged_df[timestamp], column_data], ignore_index=True,axis = 0)
                # merged_df[timestamp] = concatenated_data
                # print(concatenated_data)
                merged_data[timestamp].extend(column_data)
                max_length = max(max_length, len(merged_data[timestamp]))
                if merged_df[timestamp].dtype != column_data.dtype:
                    print(f"Data types of columns '{timestamp}' do not match.")

# Fill shorter arrays with NaN to ensure equal length
for header, values in merged_data.items():
    if len(values) < max_length:
        merged_data[header].extend([np.nan] * (max_length - len(values)))

merged_df = pd.DataFrame(merged_data)
merged_lengths = [merged_df[header].count() for header in merged_df.columns]
```

142	143	144	145	146	147	148	149	150
78,96	68,57	67,52	71,3	85,14	71,96	84,54	65,9	72,85
75,02	68,57	74,97	65,6	75,6	82,52	73,12	75,16	72,07
73,44	68,57	68,33	88,94	90,75	82,52	66,46	68,94	76,69
67,4	68,57	75,55	82,67	79,72	82,52	73,46	68,94	71,35
67,4	68,57	75,55	77,09	84,15	82,52	73,46	68,94	71,35
67,4	68,57	75,55	96,14	74	82,52	73,46	68,94	71,35
67,4	68,57	75,55	96,14	74	82,52	73,46	68,94	71,35
67,4	68,57	75,55						

Étape 2 : Normaliser toutes les données.

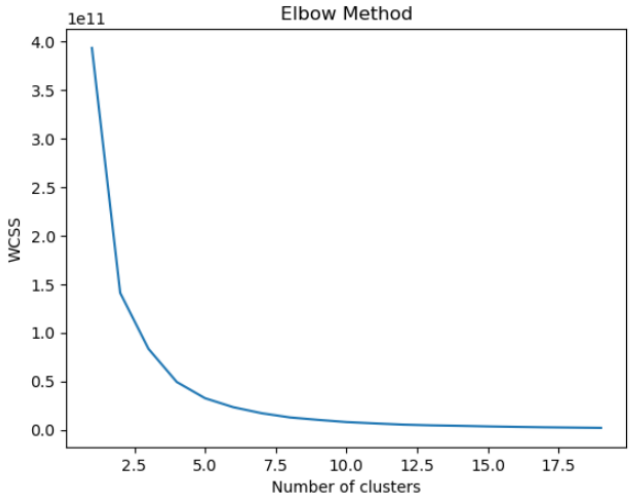
Comme nous l'avons vu ci-dessus, les fichiers varient en longueur et en échelle. Pour pouvoir les normaliser, mon tuteur a suggéré de remplacer les valeurs de chaque ligne par 5 valeurs de caractéristique de cette ligne, y compris 'Médiane','Moyenne', 'écart type','Énergie','min', 'max'.

```
                0            1            2            3  \
0   Median     59.260000    63.895000    70.390000    73.510000
1  Average     59.790000    65.671667    70.508333    72.048333
2      Std      1.185116     5.250414     0.264601     3.268386
3   Energy  21457.491600  26042.007900  29828.970500  31209.868100
4      min     59.260000    60.750000    70.390000    64.740000
5      max     62.440000    73.120000    71.100000    73.510000

            4            5            6            7            8  ...
0     83.670000    84.720000    81.100000    78.350000    75.390000  ...
1     81.493333    84.445000    80.968333    78.583333    75.081667  ...
2      4.286407     0.614919     0.294416     0.521749     0.481817  ...
3  39957.220000  42788.016900  39335.746100  37053.675000  33824.932900  ...
4     72.980000    83.070000    80.310000    78.350000    74.110000  ...
5     84.720000    84.720000    81.100000    79.750000    75.390000  ...

            989          990          991          992          993  \
0     86.800000    79.620000    84.680000    86.530000    93.020000
1     87.174444    83.673333    87.513333    88.902222    91.575556
2      3.090362     5.367224     9.911062     3.876321     2.731166
3  68480.406900  63270.304200  69811.314000  71267.678800  75541.874800
4     82.100000    79.620000    72.820000    86.530000    85.680000
5     94.870000    94.370000   103.300000    98.340000    93.020000

            994          995          996          997          998
0     93.600000    78.610000    96.510000    82.610000    99.950000
1     92.582222    83.997778    94.485556    82.534444    98.186667
2      6.105731     8.018258     3.398654     4.440308     3.397640
3  77478.730400  64079.272200  80451.639500  61484.857700  86869.489200
4     79.780000    78.390000    86.560000    74.420000    90.290000
5    103.120000   103.020000    96.510000    91.190000    99.950000

[6 rows x 1000 columns]
```

Étape 3 : Transposer les nouvelles données et appliquer PCA

PCA (Principal Component Analysis) est une technique d'analyse de données qui simplifie des ensembles de données complexes en les transformant en une forme plus gérable. Il aide à révéler les modèles et structures essentiels dans les données, ce qui en fait un outil précieux pour diverses applications parce qu'il réduit le nombre de caractéristiques dans les données de grande dimension, améliore l'efficacité et évite le surajustement

Étape 4 : Utiliser Méthode du coude (Elbow method) pour trouver le nombre de centre de cluster optimale. Ici je choisi k = 8



Étape 5: Appliquer KMeans aux nouvelles données transposées. Le modèle nous donnera une liste d'étiquettes pour chaque colonne dans les données fusionnées, les lignes avec le même numéro signifient qu'elles appartiennent au même cluster, ce qui signifie également que KMeans prédit dans ces minutes que la personne a le même sentiment.

Le modèle suivant nous donnera une liste d'étiquettes pour chaque ligne dans le données fusionnées, les lignes avec le même numéro signifient qu'elles appartiennent au même cluster, ce qui signifie également que KMeans prédit dans ces minutes que la personne a le même sentiment.

```
[7 7 7 7 0 0 0 0 7 7 7 7 7 0 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 0 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 0 7 7 7 7 7 7 7 7 7 7 7 7 5 3 5 3 3 0 0 5 5
 3 0 5 3 3 5 3 0 3 3 3 3 3 0 5 5 5 0 3 0 3 0 5 3 0 0 0 3 0 5 0 0 3 3 5 3 5
 3 3 3 3 3 0 3 3 3 0 7 0 0 3 0 3 3 3 3 3 7 3 0 7 5 0 0 0 3 0 1 5 0 0 0 3 3 3
 0 7 0 0 7 7 7 0 0 7 0 0 0 3 7 0 0 3 7 0 0 0 7 0 0 7 0 7 7 0 3 3 0 0 0 7 7
 0 7 7 7 0 0 0 7 0 0 7 0 7 7 0 7 7 0 7 7 7 7 0 7 7 7 0 7 7 0 3 7 7 7 3 0 7
 4 2 1 2 2 1 2 2 2 2 2 2 2 1 2 1 1 2 2 2 1 2 2 1 1 1 1 1 2 2 2 1 5 1 5 1 5 5 5
 1 1 5 3 5 1 1 5 5 1 5 5 5 1 3 5 5 5 1 3 3 5 1 5 5 5 1 2 5 1 5 3 5 3 5 1 0
 3 1 5 0 0 5 3 5 5 7 0 0 0 5 3 3 3 3 5 3 3 3 3 3 5 3 5 0 0 0 0 0 3 3 5 3 3
 3 5 5 3 0 3 3 5 3 5 3 3 3 3 0 3 0 3 3 0 0 3 3 3 5 3 3 3 1 1 5 0 3 0 7 7 7 0
 0 3 0 0 0 0 0 7 0 7 0 3 0 0 0 0 0 0 3 0 0 0 0 5 3 0 3 3 3 3 0 0 0 3 3 3 3
 0 0 0 0 0 0 5 0 0 0 0 0 0 3 0 0 7 0 3 3 3 3 5 3 3 3 3 6 4 2 2 4 4 6 2 2 4 6 6
 4 4 4 4 4 4 2 4 6 1 1 1 1 5 1 5 5 2 1 1 1 5 1 5 5 5 1 2 5 1 1 5 1 1 1 2 1
 2 1 1 1 1 5 2 1 1 1 1 2 1 2 1 2 1 2 1 2 1 1 1 2 1 2 2 2 2 2 2 3 3 3 0 3 5 3 0
 0 3 3 3 0 3 3 3 0 3 3 3 3 3 3 0 0 3 3 5 5 3 0 3 0 3 0 0 3 0 0 0 3 3 3 3 3 3
 3 0 0 3 3 3 0 0 0 3 0 0 0 0 0 0 3 0 0 0 0 0 0 3 0 7 0 0 0 0 0 0 0 0 7 0 0 0 7 7
 0 0 5 7 7 7 7 7 3 0 3 0 7 0 3 0 3 3 3 0 0 0 0 0 7 0 7 7 7 0 7 7 7 0 0 0 0 7
 0 0 0 0 0 0 7 7 0 7 7 7 7 7 7 7 7 7 0 0 0 0 3 5 5 5 3 5 5 5 5 5 5 3 3 5 3
 5 3 5 5 5 5 5 5 3 5 1 5 5 3 5 3 3 3 0 3 5 3 3 3 5 5 3 3 3 3 3 5 1 3 3 3 0 0 3
 0 3 0 0 3 3 3 3 5 0 0 3 3 3 3 0 0 3 3 5 3 0 3 3 3 5 5 5 5 1 5 1 3 5 1 5 5
 5 1 1 5 1 1 5 5 1 1 3 2 1 5 5 5 3 3 1 5 5 3 1 1 5 5 5 5 5 5 5 3 5 5 5 1 5
 5 3 5 5 1 5 3 3 3 5 5 5 3 5 3 3 3 3 5 3 5 3 5 5 5 3 3 5 3 3 3 3 1 1 3 3
 3 5 5 5 5 3 3 0 5 3 5 5 3 0 3 3 0 0 0 3 0 0 3 3 3 3 1 1 5 3 5 5 5 3 1 3 3 3
 0 3 1 3 3 0 3 3 3 3 3 3 0 0 5 5 0 0 0 0 0 0 0 3 5 0 3 0 3 0 0 0 5 0 0 0 0 3 3
 5 5 0 3 0 3 0 3 3 0 0 3 0 0 3 3 3 3 5 0 0 0 0 0 0 0 1 2 4 2 2 4 2 2 2 2 2
 2 1 2 2 1 2 2 2 2 4 4 4 2 2 2 1 2 1 2 1 5 1 2 2 1 2 2 2 2 2 1 1 1 2 5 1 1
 1 5 5 5 1 2 2 5 3 3 3 1 1 1 1 5 2 1 1 1 2 1 3 5 2 1 1 1 5 1 1 1 1 5 1 5 2]
```

Étape 6: Nous appliquons inversement les valeurs de la liste de étiquettes à chaque sous-fichier composant. Chaque fichier recevra une séquence d'étiquettes dont la longueur correspond à son

nombre de colonnes, et la fin d'une séquence sera le début de la suivante. Dans le graphique ci-dessous, chaque émotion correspond à un nombre différent et à une couleur différente, en la regardant, on peut dire à un certain moment dans quelle émotion se trouve cette personne.

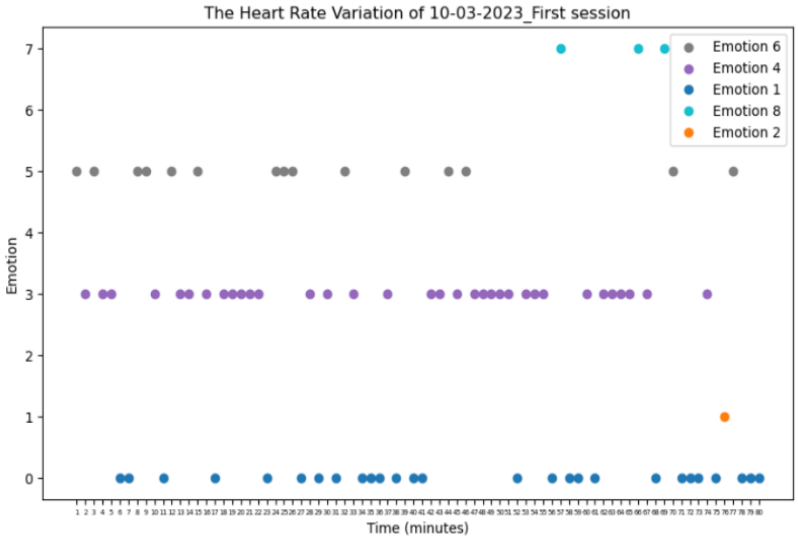


Fig . 23 : Graphique montrant les prédictions de KMeans pour chaque minute d'une session d'études

2.4.1.4 : Analyse les émotions dominantes des élèves avec des données vidéo et un oxymètre

C'est le problème dont j'ai parlé depuis le début, je vais utiliser une combinaison de données d'image et de données de l'oxymètre correspondant en termes de temps à appliquer à KMeans. Avec 2 modalités combinées, le regroupement sera probablement plus efficace que de simples images ou données oxymètre.

A propos, les étapes sont presque les mêmes qu'avec l'oxymètre de données. Cependant, pour chaque image, j'utilise FaceNet pour intégrer chaque image dans un vecteur long de 512 unités, à chaque minute, nous aurons un lot de 120 images. Nous organiserons ces 120 images dans une matrice de la forme (120,512), puis nous ferons la moyenne des colonnes afin que cette matrice revienne à la forme (512,1) , qui est la représentation picturale de toutes les images de cette personne. cette minute. Avec les données de l'oxymètre, nous utiliserons MinMaxScaler pour normaliser tous les colonnes

```
# Define a function to preprocess images
def preprocess_image(image):
    img = Image.open(image)
    img = img.resize((160, 160))  # FaceNet model input size
    img = np.array(img)
    return img


# Process each batch of 120 images from the folder
batch_size = 120
image_folder = r'E:\Projet 2023\Data\Faces_data\Micheal\23-03-2023-session-1-landmark-compare'

batch_number = 1
average_embedding_dict = {}  # Dictionary to store batch_number: average_embedding pairs

total_images_processed = 0

for batch_start in range(1, len(image_paths), batch_size):
    batch_end = batch_start + batch_size
    batch_image_paths = image_paths[batch_start:batch_end]

    batch_embeddings = []
    for image_path in batch_image_paths:
        img = preprocess_image(image_path)
        embedding = facenet(torch.tensor(img.transpose(2, 0, 1)).unsqueeze(0).float())
        batch_embeddings.append(embedding.detach().numpy().flatten())
        total_images_processed += 1

    batch_embedding_matrix = np.vstack(batch_embeddings)
    print(batch_embedding_matrix.shape)
    # Calculate the average vertically
    average_embedding = np.mean(batch_embedding_matrix, axis=0)
    print(average_embedding.shape)
    # Store the average_embedding along with the batch_number
    average_embedding_dict[batch_number] = average_embedding

    # Print progress
    print(f"Processed batch {batch_number}, Total images processed: {total_images_processed}")

    batch_number += 1
```

Ensuite, pour chaque minute, nous allons concaténer les données de 2 sources ensemble avec le clé est le batch_number

```
for batch_number, average_embedding in average_embedding_dict.items():
    normalized_columns = df.iloc[:,batch_number]

    average_embedding_with_normalized = np.concatenate([average_embedding, normalized_columns])
    average_embedding_dict[batch_number] = average_embedding_with_normalized
```

Voici un exemple de composant dans average_embedding_dict

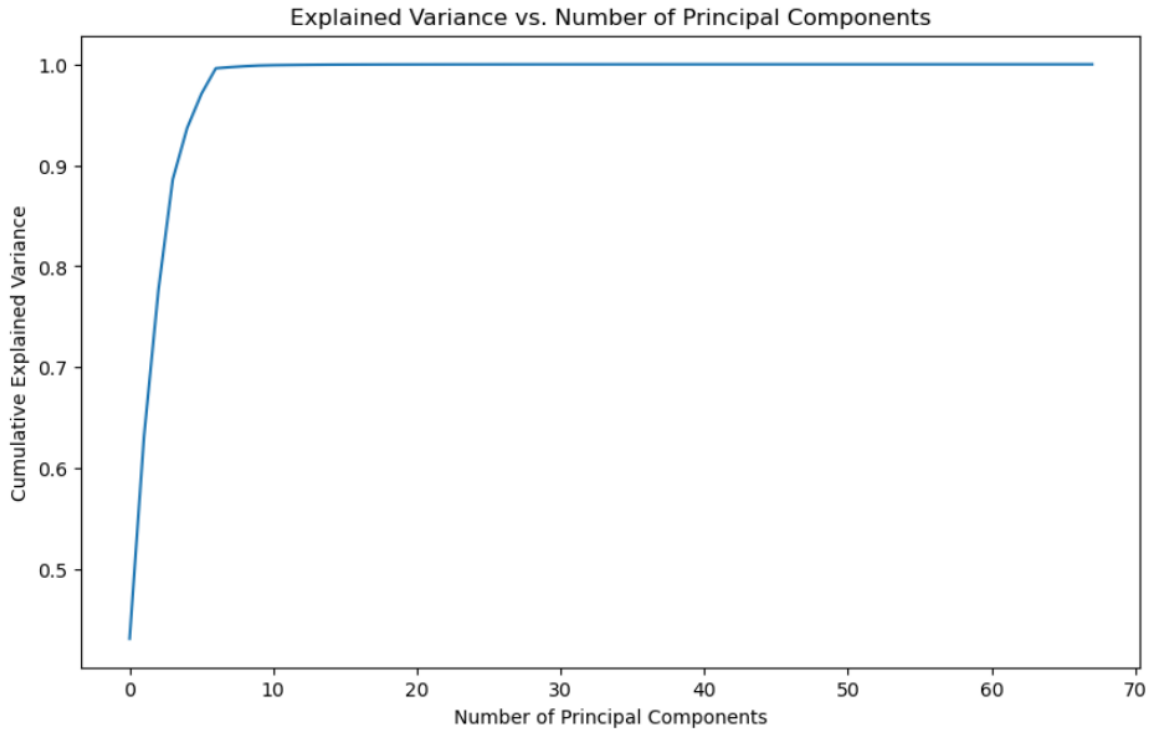
```
-3.10783796e-02 -3.33434343e-02  9.18469671e-03  3.01772989e-02
 2.07414143e-02  2.24764869e-02  1.24227991e-02  5.36103472e-02
-3.88353728e-02 -5.22844717e-02 -1.87358335e-02  1.68382842e-02
-7.19440449e-03 -5.54740475e-03 -9.34705604e-03  8.99811741e-03
 9.30806622e-03  4.65656593e-02 -3.05549689e-02  3.26116718e-02
-3.52876820e-02  6.37563765e-02  1.79372355e-03  5.53151183e-02
 2.85959635e-02 -1.16032325e-02 -3.41654196e-02 -2.71996055e-02
 3.56240198e-02 -2.49426230e-03  4.35627252e-02 -4.41086181e-02
 6.78166896e-02 -3.24771777e-02  6.31184056e-02  3.54999513e-03
 1.23504484e-02  2.95164455e-02 -3.67263518e-02  1.28882630e-02
 6.01376370e-02  2.11502481e-02 -3.38355079e-02  1.34552410e-02
 3.59138735e-02 -1.13713657e-02 -1.38615081e-02  1.99009292e-02
 1.24305394e-02 -3.54720047e-03 -6.63160756e-02  7.20874295e-02
 6.15474433e-02 -7.16706365e-02  1.90188214e-02 -4.54787984e-02
-6.19995482e-02  1.91392619e-02  2.16993205e-02 -1.69841163e-02
 1.00000000e+00  7.06756757e-01  4.06756757e-01  7.97297297e-02
 1.27477477e-01  0.00000000e+00  0.00000000e+00]
```

Après avoir obtenu les données, la prochaine chose que nous devons faire est d'appliquer PCA. Ceci est différent de ce qui précède car ici nous combinons 2 sources de données ensemble. Il faut donc tenir compte de la variance expliquée et des variances cumulées.

La variance expliquée d'une composante principale indique la part de la variance totale dans les données qu'elle capture. Lors de la combinaison de deux ensembles de données, certaines composantes principales peuvent capturer la variance des deux ensembles, tandis que d'autres peuvent capturer la variance d'un seul des ensembles

Variance cumulée : La variance cumulée montre la proportion de la variance totale expliquée par un certain nombre de composantes principales.

Pour déterminer k optimal , nous devons définir le point où l'ajout de clusters n'améliore pas significativement la variance expliquée,




```
Optimal number of components (k) for PCA: 6
```

Fig.24 : Graphique de la relation entre la variance expliquée cumulée et le nombre de composants

D'après le courbe j'ai choisi k = 6

Ensuite, nous appliquons K-Means aux données combinées, avec la mise en garde que nous devons les convertir en matrice

```
:
# Count occurrences of each label
label_counts = np.bincount(cluster_labels)
# Print the count of each label
for label, count in enumerate(label_counts):
    print(f"Cluster {label}: {count} samples")
```

```
Cluster 0: 10 samples
Cluster 1: 10 samples
Cluster 2: 9 samples
Cluster 3: 17 samples
Cluster 4: 9 samples
Cluster 5: 13 samples
```

Le résultat nous donne l'émotion numéro 4 dominante avec (17 échantillons). Bien sûr, je ne peux pas dire exactement quel est ce sentiment, mais si j'ai la feuille de réponses de mon collègue Hugo, je peux comparer l'émotion dominante de cette personne ce jour-là pour savoir qu'est ce que c'est.

En termes de classe, ma méthode ne sera probablement pas populaire car lorsque nous ne pouvons pas acquérir passivement des données d'oxymètre comme une caméra. Cependant, cette méthode peut toujours être appliquée dans les hôpitaux et les maisons de retraite pour aider les médecins à mieux surveiller les patients.

III.Conclusion

Le stage avait un sujet très intéressant et proche de la réalité. Après le stage, j'ai complété la tâche principale établie au début, qui était de construire un ensemble de procédures pour analyser les émotions des élèves à partir des données obtenues en classe en utilisant la méthode non supervisée.

Bien que ce ne soit pas la version la plus parfaite, c'est mon premier produit dans un environnement réel. Les difficultés techniques m'ont aidé à acquérir beaucoup de connaissances et d'expérience pour résoudre les problèmes qui se posent

La période de 6 mois à l'ESAIP sera un bagage important pour m'aider à poursuivre une carrière dans l'informatique en général et le Deep Learning en particulier.

Encore une fois, je tiens à remercier M. Pejman Rasti, M. Hidane Moncef pour son aide et ses conseils au cours du temps passé.

Reference :

[1] Estimation of heart rate and respiratory rate from the seismocardiogram under resting state

Yue-Der Lina,b,*, Ya-Fen Jhoub,c a Department of Automatic Control Engineering, Feng Chia University, Taichung, Taiwan b Master's Program of Biomedical Informatics and Biomedical Engineering, Feng Chia University, Taichung Taiwan c ACME Portable Corp, New Taipei City, Taiwan

[2] Evaluation of Haar Cascade Classifiers for Face Detection

April 2012

Conference: ICDIP: International Conference on Digital Image ProcessingAt: Venice, ItalyVolume: 6

[3] FaceNet: A Unified Embedding for Face Recognition and Clustering

Florian Schroff, Dmitry Kalenichenko, James Philbin

ESAIP, Saint-Barthélemy-d'Anjou (49), classement - L'Etudiant (letudiant.fr)

The Influences of Emotion on Learning and Memory - PMC (nih.gov)

Face Detection with Haar Cascade. Exploring a bit older algorithm which… | by Girija Shankar Behera | Towards Data Science

(99+) Face recognition using Histograms of Oriented Gradients | Oscar Deniz - Academia.edu

17 types of similarity and dissimilarity measures used in data science. | by Mahmoud Harmouch | Towards Data Science