Chapitre III

Méthode proposée

Sommaire

1 Introduction

L'objectif de ce chapitre est de présenter les étapes de déploiement des modèles proposées dans le cadre d'un système de reconnaissance des expressions faciales et les différentes étapes de réalisation. Nous nous sommes intéressés à l'utilisation de réseau neuronal convolutif. Nous commençons tout d'abord par la présentation de deux piste de modélisation utilisé puis l'architecture de chaque modèle, ainsi que l'étapes de la réalisation du modèle. Nous poursuivons ce chapitre par la présentation des hyperparamètres qui nous intéresse et nous clôturons par les méthodes d'optimisation de ces hyperparamètres. Toutes les questions qui concernent le réseau CNN utilisé dans notre travail seront débattues en détail dans ce chapitre.

2 Conception détaillé des systèmes proposés

Les émotions visuelles ont été classées en deux classes (positives et négatives), trois polarités : positive, négative et neutre, ou les sept catégories de base. Avec la croissance rapide des médias sociaux, nous communiquons nos sentiments en téléchargeant des vidéos, des sons, des photos et des mots sur des blogs (ou microblogs ou Chat). Les images vives, par exemple, indiquent souvent des sentiments agréables (ou positifs), tandis que les images sombres symbolisent souvent des émotions terribles (ou négatives). Outre les mots et les sons, les images véhiculent une riche sémantique émotionnelle en raison de leur abondance d'informations visuelles, qui peuvent être utilisées pour effectuer diverses tâches. Dans ce contexte, nous présenterons deux modèles d'analyse des expressions faciales, un modèle CNN basé sur l'apprentissage par transfert et le second modèle est un CNN construit à partir de zéro dans Keras. Dans ce travail, nous avons utilisé deux pistes de modélisation différentes (From scratch et l'apprentissage par transfert). Nous allons présenter deux modèles d'analyse des expressions faciales, un modèle CNN basé sur l'apprentissage par transfert et testé sur VGG-16 et VGG-19 et le second modèle est un CNN construit à partir de zéro dans Keras. Dans la suite, pour les deux modèles proposés, nous allons mentionner en détail chacune des étapes de conception de chaque modèle.

2.1 Détection du visage

Le problème de la détection des visages consiste à identifier la présence de visages dans une image et à déterminer l'emplacement et l'échelle des visages. L'efficacité des systèmes de FER dépend essentiellement de la méthode utilisée pour localiser le visage dans l'image. Dans les méthodes proposées, nous avons utilisé l'algorithme de détection d'objets en cascade de Haar utilisé pour identifier les visages dans une image ou une vidéo en temps réel. L'algorithme utilise les caractéristiques de détection des bords ou des lignes proposées par Viola et Jones dans leur article de recherche. Nous avons utilisé la bibliothèque python d'apprentissage automatique connue sous le nom d'OpenCV, qui dispose de divers classificateurs préformés. Il existe de nombreux classificateurs Haar en cascade pour la détection des yeux, de

la bouche et du nez, mais pour cette technologie, nous avons uniquement utilisé le classificateur Haar en cascade pour le visage frontal. Pour détecter le visage dans la vidéo, nous avons utilisé le module detect-Multi-Scale d'OpenCV. Cela crée un rectangle vert avec des coordonnées (x, y, w, h) autour du visage détecté dans l'image

Principe de la méthode de Viola et Jones : La méthode de Viola et Jones consiste à balayer une image en utilisant une fenêtre de détection de taille initiale 24px par 24px (dans l'algorithme original) et à déterminer si un visage est présent. Lorsque l'image a été entièrement balayée, la taille de la fenêtre est augmentée et le balayage reprend jusqu'à ce que la fenêtre ait la taille de l'image. L'augmentation de la taille de la fenêtre se fait par un facteur multiplicatif de 1,25. Le balayage, quant à lui, consiste simplement à déplacer la fenêtre d'un pixel. Ce décalage peut être modifié pour accélérer le processus, mais un décalage d'un pixel garantit une précision maximale. Cette méthode est une approche basée sur l'apparence, qui consiste à balayer l'image entière en calculant un certain nombre de caractéristiques dans des zones rectangulaires qui se chevauchent. Elle a la particularité d'utiliser des caractéristiques très simples, mais très nombreuses.

La bibliothèque OpenCV présente une implémentation de cet algorithme sous le nom de "Haar cascade detector". Le point fort de cette méthode est la rapidité de détection qui la rend capable de fonctionner en temps réel et de répondre aux exigences du traitement vidéo. Cependant, elle présente certaines limites telles que la difficulté de détecter plusieurs vues d'un même objet simultanément, le temps nécessaire à la phase d'apprentissage des cascades est relativement important et le nombre d'échantillons d'apprentissage est important.

Comme nous traitons des séquences vidéo, la phase de détection implique implicitement la phase de suivi des visages dans la scène, puisque nous traitons la séquence vidéo image par image. Cette étape se décompose en trois tâches : la détection du visage par la méthode, l'extraction des traits ou points caractéristiques du visage et le suivi des mouvements du visage dans la scène.

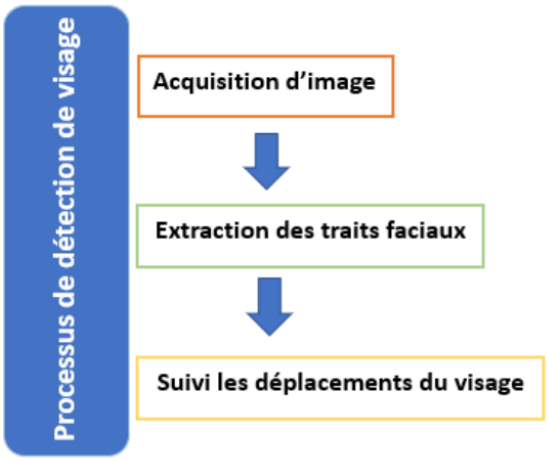


FIGURE III.1 – *Processus de détection du visage*

2.2 Extraction des caractéristiques faciales

Une fois le visage détecté dans l'image, le système lance le processus d'extraction des caractéristiques qui va convertir les données des pixels à des représentations et configuration plus réduite et optimal pour que la représentation extraite soit utilisé dans le processus de la classification , cette étape réduit les dimensions de l'image en entrée en gardant les données les plus utiles pour la classification. L'étape d'extraction représente le coeur du système de reconnaissance, on extrait de l'image les informations qui seront sauvegardées en mémoire pour être utilisées plus tard dans la phase de décision (classification).

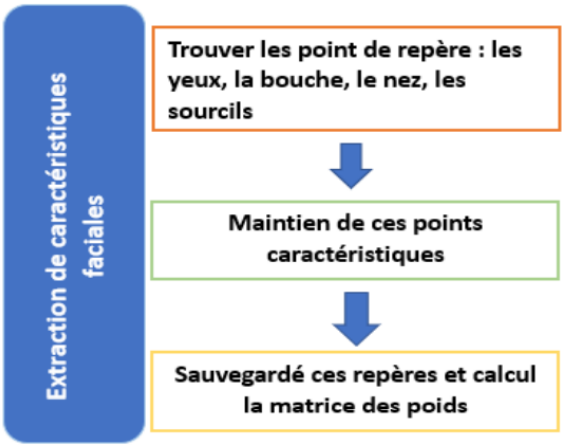


FIGURE III.2 – *Processus de l'étape d'extraction de caractéristiques faciales*

La détection de ces points est implémentée dans la librairie dlib [22] utilisé sous le langage Python. Elle permet de produit 68 point 2D de coordonnées (x, y) qui cartographient des structures faciales spécifiques. Ces points sont stockés dans un

tableau indexé. Donc la figure suivante présente les indices de chaque point parmi les 68 points.

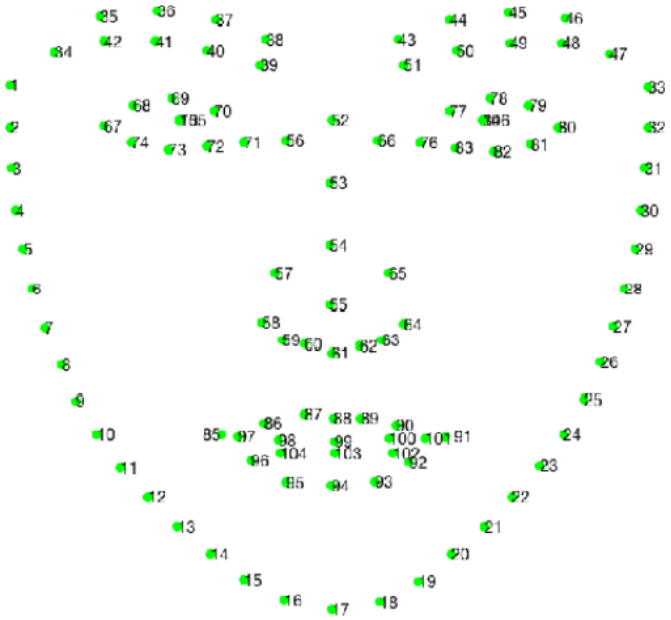


FIGURE III.3 – *Le résultat de détection des points caractéristiques à partir du visage en utilisant dlib[22].*

3 Classification des images

3.1 Modèles CNN basés sur l'apprentissage par transfert

L'apprentissage par transfert est l'un des outils les plus pratiques à utiliser si l'on travaille sur un problème de classification d'images. Il s'agit d'un apprentissage qui peut être transféré. En traitement d'images, il est couramment utilisé pour améliorer l'efficacité de l'apprentissage et de la formation.
Le transfert d'informations de domaines connexes (domaines sources) vers d'autres domaines (domaines cibles) est l'un des principes de l'apprentissage par transfert [6v]. Le VGG est un algorithme connu en vision par ordinateur très souvent utilisé par l'apprentissage par transfert pour éviter de devoir le réentraîner et résoudre des problèmes similaires sur lesquels le VGG a déjà été entraîné. Il existe de nombreux autres algorithmes du même type que VGG comme ResNet ou Xception disponibles dans la bibliothèque Keras.
Dans ce travail, nous nous concentrerons sur l'architecture VGG16 et VGG19.

3.1.1 Modèle VGG16

Il s'agit d'un modèle de réseau neuronal convolutif (CNN) proposé par Karen Simonyan et Andrew Zisserman de l'université d'Oxford. L'idée du modèle a été proposée en 2013, mais le modèle réel a été soumis au défi ImageNet de l'ILSVRC

en 2014. VGG16 a été identifié comme le modèle le plus performant sur le jeu de données ImageNet. Passons en revue l'architecture réelle de cette configuration.

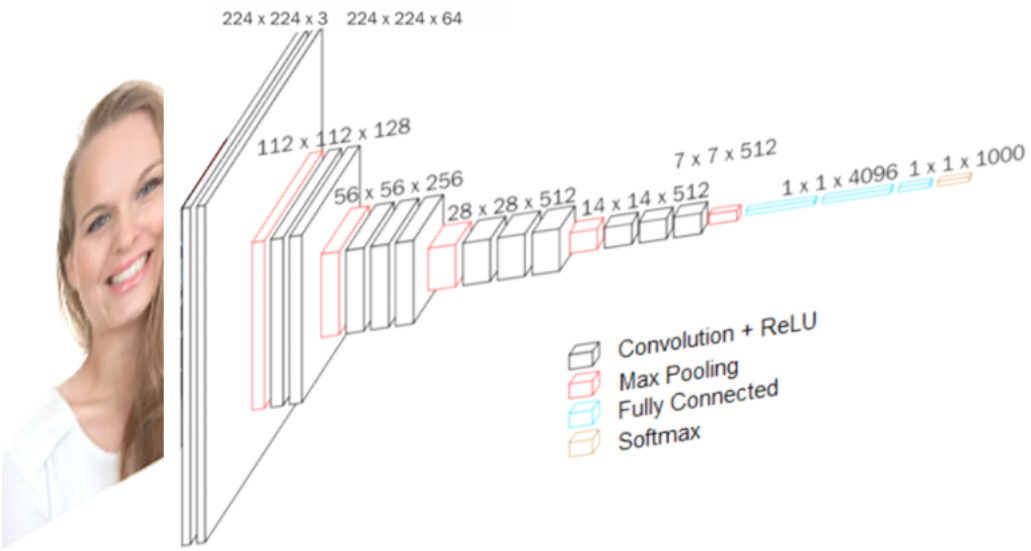


FIGURE III.4 – *Architecture standard du modèle VGG-16*

L'entrée de l'une des configurations du réseau est considérée comme une image de taille fixe 224 x 224 avec trois canaux - R, G et B. Le seul prétraitement effectué consiste à normaliser les valeurs RVB de chaque pixel. Pour ce faire, on soustrait la valeur moyenne de chaque pixel.

L'image passe par la première pile de 2 couches de convolution de la très petite taille réceptive de 3 x 3, suivie par des activations ReLU. Chacune de ces deux couches contient 64 filtres. Le pas de convolution est fixé à 1 pixel et le padding à 1 pixel. Cette configuration préserve la résolution spatiale et la taille de la carte d'activation de sortie est la même que les dimensions de l'image d'entrée. Les cartes d'activation sont ensuite passées par un pooling spatial maximal sur une fenêtre de 2 x 2 pixels, avec un stride de 2 pixels. Cela divise par deux la taille des activations.Ainsi, la taille des activations à la fin de la première pile est de 112 x 112 x 64.

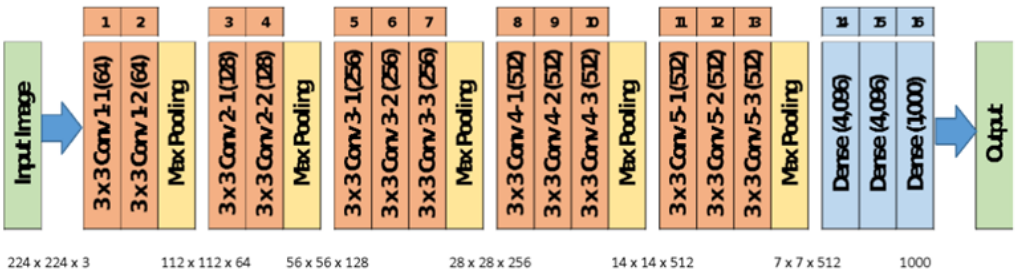


FIGURE III.5 – *Les couches du modèle VGG-16*

Les activations passent ensuite par une deuxième pile similaire, mais avec 128 filtres contre 64 dans la première. Par conséquent, la taille après la deuxième pile devient 56 x 56 x 128. Elle est suivie par la troisième pile avec trois couches convolutionnelles et une couche de pool max. Le nombre de filtres appliqués ici est de 256, ce qui donne la taille en sortie de la pile 28 x 28 x 256. Elle est suivie de deux piles de trois couches convolutionnelles, chacune contenant 512 filtres. La sortie à la fin de ces deux piles sera de 7 x 7 x 512. Les piles de couches convolutionnelles sont suivies de trois couches entièrement connectées avec une couche d'aplatissement entre les deux. Les deux premières ont 4 096 neurones chacune, et la dernière couche entièrement connectée sert de couche de sortie et possède 1 000 neurones correspondant aux 1 000 classes possibles pour le jeu de données ImageNet. La couche de sortie est suivie par la couche d'activation Softmax utilisée pour la classification catégorielle.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 512)	12845568
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 512)	262656
dense_3 (Dense)	(None, 4)	2052

Total params: 27,824,964
Trainable params: 27,786,244
Non-trainable params: 38,720

FIGURE III.6 – *Architecture du Modèle VGG-16*

3.1.2 Modèle VGG19

VGG19 est un réseau neuronal convolutif à 19 couches avec 16 couches convolutifs et trois couches entièrement connectées. La collection ImageNet, qui comprend un million d'images dans 1000 catégories, a été utilisée pour former VGG19.

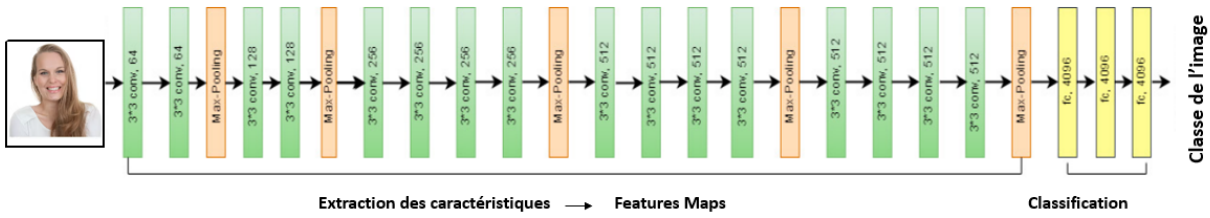


FIGURE III.7 – *Les couches du modéle VGG-19*

Le modèle ne nécessite qu'un prétraitement spécifique qui consiste à soustraire la valeur RVB moyenne, calculée sur l'ensemble d'apprentissage, de chaque pixel. Pendant l'apprentissage du modèle, l'entrée de la première couche de convolution est une image RVB 224x224. Pour toutes les couches de convolution, le noyau de convolution est de taille 3 3 : la plus petite dimension pour capturer les notions de haut, bas, gauche/droite et centre. Il s'agissait d'une spécificité du modèle au moment de sa publication.
Jusqu'au VGG16, de nombreux modèles étaient orientés vers des noyaux de convolution plus grands (taille 11 ou 5 par exemple). Rappelons que ces couches sont destinées à filtrer l'image en ne gardant que les informations discriminantes comme les formes géométriques atypiques.
Ces couches de convolution sont accompagnées d'une couche de Max-Pooling, chacune de taille 2 2, afin de réduire la taille des filtres lors de l'apprentissage. En dehors des couches de convolution et de pooling, nous avons 3 couches de neurones entièrement connectées. Les deux premières sont composées de 4096 neurones et la dernière de 1000 neurones avec une fonction d'activation softmax pour déterminer la classe de l'image
En raison de l'utilisation de nombreux filtres 3x3 dans chaque couche convolutive, il s'agit d'une approche largement courante pour la classification des images [7]. Dans cette étude, nous avons utilisé les deux jeux de données CK+, FER2013.

— **Méthodologie**
Nous avons adopté un pipeline d'approche cohérent pour toutes les activités d'apprentissage automatique et d'apprentissage profond. La collecte de données, le prétraitement, l'extraction de caractéristiques, le réglage du modèle et la catégorisation des sentiments font tous partie du flux de travail. Dans la figure suivante, nous pouvons voir l'ensemble du processus proposé dans ce travail.

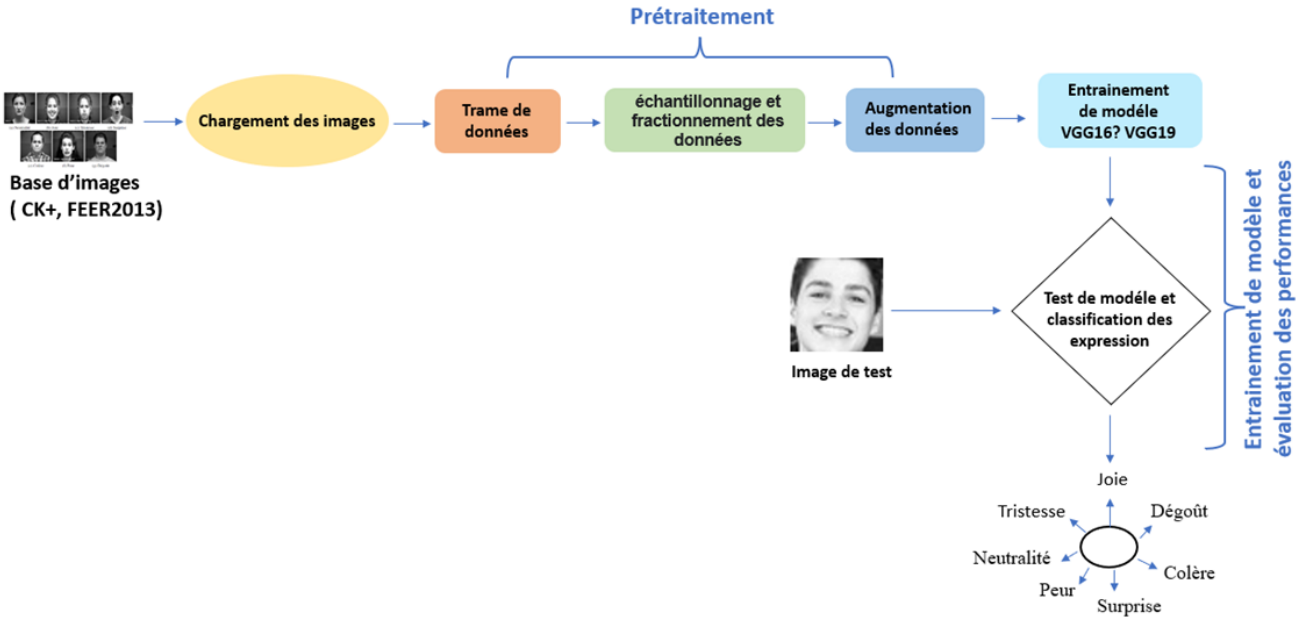
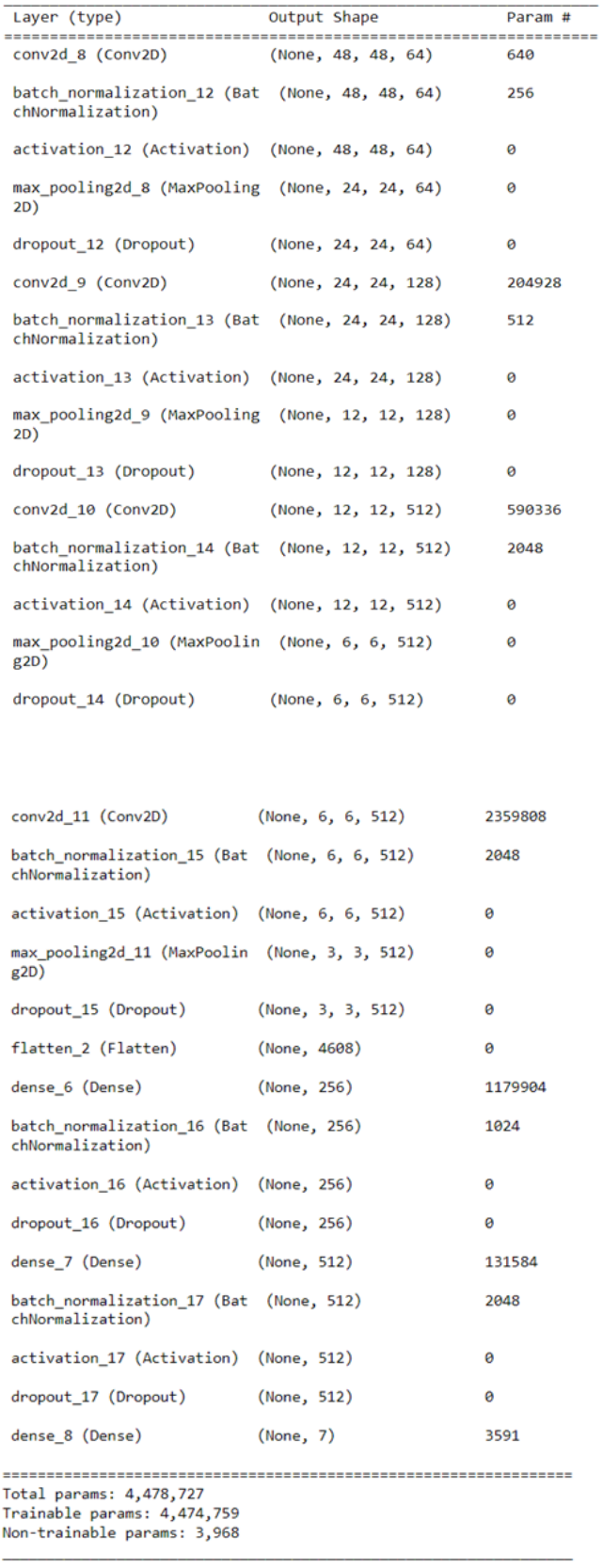


FIGURE III.8 – *Flux de travail d'analyse des sentiments d'image proposé*

Ceci a été fait en redimensionnant et en retournant chaque image pour qu'elle corresponde aux exigences d'entrée du modèle proposé basé sur VGG-19 dans l'étape de prétraitement. Pour ce faire, nous avons utilisé TensorFlow ImageDataGenerator [23] de l'API Keras de TensorFlow. Parmi les fonctions de la classe ImageDataGenerator figurent le redimensionnement et le retournement horizontal. Le redimensionnement de l'image s'effectue en spécifiant une valeur pour l'option de redimensionnement ici. Ici, les valeurs maximale et minimale des pixels sont maintenant respectivement 255 et 0, puisque nous avons réduit les images dans la plage 0-255.

Ces images ont ensuite été tournées horizontalement et mises à l'échelle à 48x48 pixels. Les valeurs de pixel vont de zéro à 255 pour chaque image de l'ensemble de données d'apprentissage, qui ont toutes la même taille (48 x 48 x 3). Le flux de fonctions du bloc de données de cette instance d'ImageDataGenerator accepte un bloc de données d'entrée panda. De même, seul le redimensionnement est effectué sur l'ensemble de test car nous ne voulons pas que les images soient renvoyées.

Ensuite, les données prétraitées sont divisées en deux groupes avec une distribution de 80% et 20%. La méthode d'échantillonnage stratifié est utilisée pour s'assurer que le même nombre de classes est échantillonné pour les ensembles de test et d'apprentissage. 80 % des données sont utilisées pour l'apprentissage et 20 % pour le test. La figure suivante présente l'architecture de modèle VGG-19 utilisé.



Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 48, 48, 64)	640
batch_normalization_12 (BatchNormalization)	(None, 48, 48, 64)	256
activation_12 (Activation)	(None, 48, 48, 64)	0
max_pooling2d_8 (MaxPooling2D)	(None, 24, 24, 64)	0
dropout_12 (Dropout)	(None, 24, 24, 64)	0
conv2d_9 (Conv2D)	(None, 24, 24, 128)	204928
batch_normalization_13 (BatchNormalization)	(None, 24, 24, 128)	512
activation_13 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_9 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_13 (Dropout)	(None, 12, 12, 128)	0
conv2d_10 (Conv2D)	(None, 12, 12, 512)	590336
batch_normalization_14 (BatchNormalization)	(None, 12, 12, 512)	2048
activation_14 (Activation)	(None, 12, 12, 512)	0
max_pooling2d_10 (MaxPooling2D)	(None, 6, 6, 512)	0
dropout_14 (Dropout)	(None, 6, 6, 512)	0
conv2d_11 (Conv2D)	(None, 6, 6, 512)	2359808
batch_normalization_15 (BatchNormalization)	(None, 6, 6, 512)	2048
activation_15 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_11 (MaxPooling2D)	(None, 3, 3, 512)	0
dropout_15 (Dropout)	(None, 3, 3, 512)	0
flatten_2 (Flatten)	(None, 4608)	0
dense_6 (Dense)	(None, 256)	1179904
batch_normalization_16 (BatchNormalization)	(None, 256)	1024
activation_16 (Activation)	(None, 256)	0
dropout_16 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 512)	131584
batch_normalization_17 (BatchNormalization)	(None, 512)	2048
activation_17 (Activation)	(None, 512)	0
dropout_17 (Dropout)	(None, 512)	0
dense_8 (Dense)	(None, 7)	3591

Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968

FIGURE III.9 – *Architecture de Modèle VGG-19*

3.2 Mise en œuvre du modèle CNN from scratch

3.2.1 prétraitement des données

Cette section décrit les différents modules du notre système de reconnaissance.
— **Module de détection et conversion des images au niveaux de gris :**
Pour détecter les visages, nous avons aussi utilisé la méthode populaire, proposée par Paul Viola et Michael Jones dans leur article "Rapid detection of objects using a cascade of simple functions". Dans notre système, nous avons utilisé haarcascade-eye.xml de la bibliothèque OpenCV qui fournit la méthode Haar Cascade. L'interface utilisateur graphique (GUI) a été écrite dans un langage de programmation python pour accepter la livraison d'images et de vidéos en direct. Le système convertit ensuite l'image d'entrée en niveaux de gris 48*48 après la détection du visage par le classificateur Haar Cascade. Après cela, l'image recadrée est passée dans le modèle proposé pour être classée en 7 émotions distinctes.

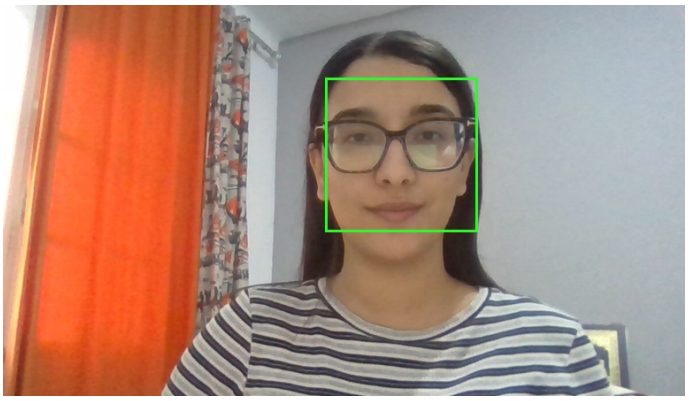


FIGURE III.10 – *Detection de visage et dessin de rectangle englobant dans chaque frame*

— **Répartition des données en base d'apprentissage et de test**
À l'aide de la bibliothèque Scikitplot, la fonction de division de l'ensemble de formation a été utilisée pour diviser l'ensemble de données en plusieurs tailles de 7 ensembles de formation et 7 ensembles de test. Nous avons considéré des tailles d'ensembles d'entraînement allant de 20% à 80%.
— **Normalisation des données**
Les réseaux neuronaux étant très sensibles aux données non normalisées, nous avons normalisé les résultats pour obtenir de meilleurs résultats. Cela a permis au réseau neuronal de fonctionner plus rapidement avec une stabilité de couche beaucoup plus grande. L'objectif principal de la normalisation des images avec 255 est de minimiser les gradients explosifs causés par la large gamme de pixels [0, 255], ainsi que d'améliorer les performances de convergence.

L'étape suivante de ce processus est la normalisation des lots. En retirant la moyenne du lot et en la divisant par l'écart type du lot, la normalisation du lot empêche la sortie d'une couche précédente. Les sorties sont mises à l'échelle pour entraîner le réseau plus rapidement et accélérer le processus d'apprentissage. Elle

élimine également les problèmes causés par une initialisation incorrecte des paramètres. Elle régule les entrées des fonctions d'activation, ce qui les rend réalisables et donne globalement de meilleurs résultats. Avec un effect de régulation, il ajoute du bruit pour minimiser l'overfitting. Les sorties des couches sont mises à l'échelle pour avoir une moyenne de 0 et une variation de 1.
Comme il s'agit d'un modèle CNN, nous avons utilisé les abandons à intervalles réguliers pour les genres. Dans le processus de normalisation par lots, nous avons choisi l'unité linéaire exponentielle (ELU) comme fonction d'activation, qui est une fonction d'activation basée sur ReLU qui defines la fluidité de la fonction lorsque les entrées sont négatives. Nous nous sommes assurés d'utiliser moins de dropout car le dropout lui-même ajoute habituellement du bruit. Nous avons fixé la valeur de remplissage à 'SAME' pendant l'entraînement du modèle car la taille de l'entrée est la même que celle de la sortie.

— **Arrêt précoce et ReduceLROnPlateau :**
L'arrêt précoce est une forme de régularisation permettant d'éviter l'overfitting lors de l'apprentissage d'un modèle. Nous avons utilisé cette technique pour éviter le surajustement de l'ensemble de données lors de l'entraînement de notre modèle DL. notre modèle DL. La technique d'overfitting utilise des règles pour fournir des indications concernant nombre d'itérations qui peuvent être exécutées avant que le modèle ne commence à sur adapter les ensembles de données. Les paramètres importants du rappel de l'arrêt précoce que nous avons utilisé lors de la formation du modèle sont les suivants :
  — **Monitor :** C'est la quantité à surveiller. Nous avons défini ce paramètre à test accuracy par défaut pour surveiller cette quantité. test accuracy est la précision de test obtenue par le modèle entraîné après avoir testé le modèle.
  — **Verbose :** Ce paramètre nous donne le choix quant à la façon dont nous voulons visualiser la sortie du réseau neuronal pendant sa formation. Nous avons fixé la valeur de ce paramètre à 1 afin de rendre la sortie visible à chaque mise à jour. Si nous le réglons sur 0, rien ne sera affiché.
  — **Restore best weights :**Ce paramètre indique s'il faut utiliser le poids du meilleur modèle ou le poids de la dernière époque. Nous avons défini ce paramètre à True qui prend le poids du meilleur modèle comme valeur.

ReduceLROnPlateauest utilisé pour réduire le taux d'apprentissage lorsqu'une métrique a cessé de montrer une amélioration. Pour que le modèle montre des progrès, le taux d'apprentissage est généralement réduit par un facteur de 2 à 10 lorsque l'apprentissage du modèle montre une stagnation. Ce planificateur lit la quantité de métriques, et s'il n'y a pas d'amélioration pendant un nombre "patient" d'époques, alors le taux d'apprentissage est réduit. Les paramètres importants de l'ordonnanceur ReduceLROnPlateau que nous avons utilisés lors de la formation du modèle sont les suivants :

— factor : C'est la valeur par laquelle le taux d'apprentissage sera réduit. Nous

avons fixé la valeur de ce paramètre à 0,5.
— patience : C'est le nombre d'époques sans amélioration après lequel l'entraînement s'arrête. Nous avons fixé la valeur de ce paramètre à 7.
— min_lr : Il indique la limite inférieure du taux d'apprentissage.Nous avons fixé sa valeur à 0.001.

3.2.2 Architecture du modèle CNN proposé :

Comme illustré sur la figure suivante, la méthode de CNN (Convolution neural network) proposée est composée par quatre couches de convolution , une couche flatten et deux couches entièrement connectés et une dropout pour faire l'extraction des caractéristique. Un bon classificateur nécessite beaucoup d'ensembles de données d'entraînement pour atteindre un bon résultat.

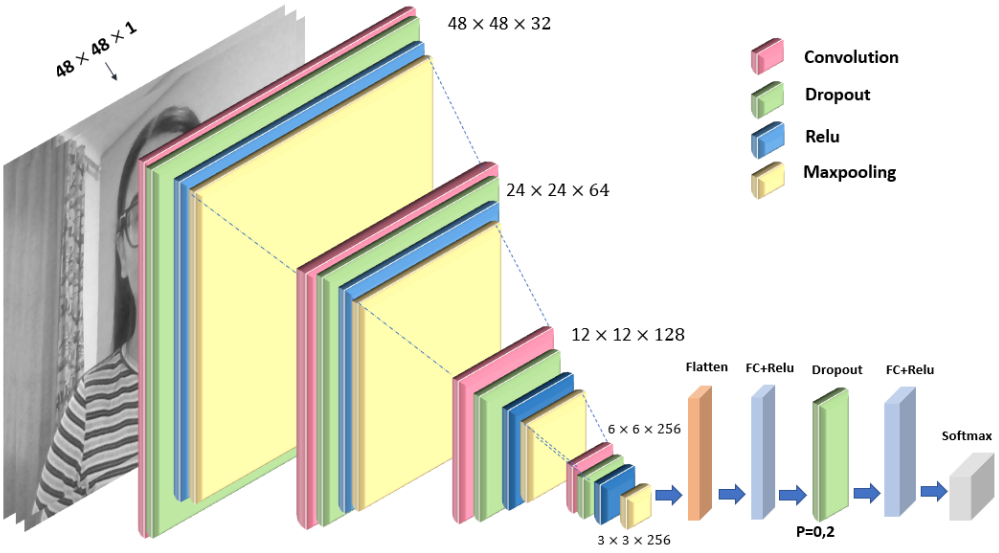


FIGURE III.11 – *Architecture en 3D du modèle CNN proposé*

Le modèle de classification des émotions a été écrit dans le langage de programmation python avec Keras, Tensorflow, Bibliothèques NumPy, OpenCV et Matplotlib. Keras fournit la fonction d'activation, les optimiseurs, les couches, l'abandon, le lot normalisation, etc. Le Tensorflow a été utilisé comme système backend pour accepter les entrées d'un tableau multidimensionnel, qui sont les pixels des images entraînées. Il est développée selon l'architecture suivante :

1. Couche d'entrée : est le point de départ de notre système. Elle présente les données (Image niveau de gris). de traitement sous la forme dés images de taille (48x48) en niveau de gris. Le choix ici est dicté par la base de données retenue et des considérations matérielles.
2. Une Dropout : pour désactiver temporairement certains neurones dans le réseau, ainsi que toutes ses connexions entrantes et sortantes, c'est une technique permettant de réduire l'overfitting lors de l'entraînement du modèle.

3. Couche de convolution :une couche prend comme entrée une image de taille 48 x48 passe par différentes filtres aide dans l'extraction des caractéristiques pertinentes pour cela nous avons utilisés cette couche 3 fois avec des filtres de taille (3 x 3), un pas et zéro rembourrage de 1. Chacune de nos couches de convolution est suivie d'une fonction d'activation linéaire rectifier (relu). Une couche de convolution est introduit par la fonction Conv2D du modèle séquentiel d'un réseau convolutif dans notre environnent :

$$\boxed{Conv2D(filters, kernelsize, activation, padding, stride)} \quad \text{(III.1)}$$

— **Filtres :** c'est le nombre de filtres de convolution. —**Kernel size :** un entier ou tuple / liste de 2 entiers, spécifiant la taille (hauteur et la largeur) du filtre de convolution 2D. —**Activation :** fonction d'activation à utiliser. Divers choix sont disponibles (relu, elu,..), Toutes les fonctions d'activation sur ces nœuds sont ReLU.

4. **Couche d'agrégation spatiale :** c'est l'étape qui, permet de réduire la taille spatiale des cartes caractéristiques locale. Nous avons utilisés le pooling de taille (2 x 2) et un pas de 2 de type max après chaque couche de convolution qui est définit par :

$$\boxed{MaxPooling(poolsize, stride).} \quad \text{(III.2)}$$

— **Pool size** = la taille de fenêtre de max pooling. —**Stride** = facteur par lequel réduire. Par exemple 2 divisera à la moitié son entrée

5. **Couche entièrement connectée :** c'est là dernière couche qui contient le raisonnement de haut niveaux, où les données sont, sous la forme d'un vecteur de dimension (n x 1). Dans cette couche, nous avons utilisés deux couches entièrement connectées avec une fonction d'activation relu, où cette couche définit par :

$$\boxed{Dense(numclasse, activation)} \quad \text{(III.3)}$$

— **Num classe :** le nombre de classe de classification 7 dans notre cas.
— **Activation :** fonction d'activation à utiliser. Dans cette couche nous avons utilisé la fonction softmax.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 48, 48, 1)]	0
conv2d (Conv2D)	(None, 48, 48, 32)	320
dropout (Dropout)	(None, 48, 48, 32)	0
activation (Activation)	(None, 48, 48, 32)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18496
dropout_1 (Dropout)	(None, 24, 24, 64)	0
activation_1 (Activation)	(None, 24, 24, 64)	0
max_pooling2d_1 (MaxPooling 2D)	(None, 12, 12, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73856
dropout_2 (Dropout)	(None, 12, 12, 128)	0
activation_2 (Activation)	(None, 12, 12, 128)	0
max_pooling2d_2 (MaxPooling 2D)	(None, 6, 6, 128)	0
conv2d_3 (Conv2D)	(None, 6, 6, 256)	295168
dropout_3 (Dropout)	(None, 6, 6, 256)	0
activation_3 (Activation)	(None, 6, 6, 256)	0
max_pooling2d_3 (MaxPooling 2D)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 128)	295040
dropout_4 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 3)	387

Total params: 683,267
Trainable params: 683,267
Non-trainable params: 0

FIGURE III.12 – *Architecture de Modèle proposé*

3.3 Optimisation empirique des hyperparamètres

L'objectif de cette section est de trouver le réseau CNN Structure optimal à encoder comme un ensemble d'hyperparamètres. Nous avons utilisé la méthode d'optimisation par recherche ainsi que la méthode d'optimisation par Grid Search Par conséquent, nous définissons les hyper paramètres suivants comme un ensemble de paramètres structurels. Le principe de cette méthode est de tester différentes combinaisons de valeurs des hyperparamètres.

3.3.1 Les hyperparamètres

— **Le taux d'apprentissage :**La quantité de mise à jour des pondérations pendant l'entraînement est appelée taille de pas ou «Taux d'apprentissage» ou Learning rate en anglais. Le défi de la formation des réseaux de neurones d'apprentissage profond implique de sélectionner avec soin le taux d'apprentissage.
C'est un hyperparamètre qui contrôle le degré de modification du modèle en réponse à l'erreur estimée chaque fois que les poids du modèle sont mis à jour. Le choix du taux d'apprentissage est difficile car une valeur trop petite peut entraîner un long processus d'entraînement qui pourrait rester bloqué, tandis qu'une valeur trop grande peut entraîner l'apprentissage trop rapide d'un ensemble de poids sous- optimal ou un processus d'entraînement instable. Le taux d'apprentissage peut être l'hyperparamètre le plus important lors de la configuration d'un réseau de neurones. Plus précisément il est configurable utilisé dans la formation des réseaux de neurones qui a une petite valeur positive, souvent comprise entre 0,0 et 1,0.

— **Le Dropout :**Les réseaux de neurones d'apprentissage en profondeur sont susceptibles de suradapter rapidement à un ensemble de données d'apprentissage avec peu d'exemples. Les ensembles de réseaux de neurones avec différentes configurations de modèles sont connus pour réduire le surajustement, mais nécessitent des dépenses de calcul supplémentaires pour la formation et la maintenance de plusieurs modèles.

Dropout fonctionne en supprimant de manière probabiliste, ou en « abandonnant », les entrées d'une couche, qui peuvent être des variables d'entrée dans l'échantillon de données ou des activations d'une couche précédente. Cela a pour effet de simuler un grand nombre de réseaux avec une structure de réseau très différente et, à son tour, de rendre les nœuds du réseau généralement plus robustes aux entrées. Keras prend en charge la régularisation des dropout. La forme la plus simple d'abandon dans Keras est fourni par une couche centrale d'abandon. Une fois créé, le taux de dropout peut être spécifié à la couche comme la probabilité de mettre à zéro chaque entrée de la couche.

— **Nombre d'époque :**En termes de réseaux de neurones artificiels, une époque fait référence à un cycle à travers l'ensemble de données d'entraînement complet. Habituellement, la formation d'un réseau de neurones prend plus que quelques époques. En d'autres termes, si nous alimentons un réseau de neurones avec

les données d'entraînement pour plus d'une époque dans différents modèles, nous espérons une meilleure généralisation lorsqu'on reçoit une nouvelle entrée "invisible" (données de test). Une époque est souvent confondue avec une itération. Les itérations correspondent au nombre de lots ou d'étapes dans les paquets partitionnés des données d'apprentissage, nécessaires pour terminer une époque.

Heuristiquement , une motivation est que (en particulier pour les ensembles d'entraînement volumineux mais finis), cela donne au réseau une chance de voir les données précédentes pour réajuster les paramètres du modèle afin que le modèle ne soit pas biaisé vers les derniers points de données pendant l'entraînement.

— **Batch_size :** Le Batch_size définit le nombre d'échantillons qui seront propagés sur le réseau. Par exemple, supposons que nous avons de 1050 échantillons d'apprentissage et que souhaiterons définir un batch_size égal à 100. L'algorithme prend les 100 premiers échantillons (du 1er au 100e) de l'ensemble de données d'apprentissage et entraîne le réseau. Ensuite, il prend les 100 secondes échantillons (du 101e au 200e) et forme à nouveau le réseau.

Nous pouvons continuer à faire cette procédure jusqu'à ce que nous ayons propagé tous les échantillons à travers le réseau. Un problème peut survenir avec le dernier ensemble d'échantillons. Dans notre exemple, nous avons utilisé 1050 qui n'est pas divisible par 100 sans reste. La solution la plus simple consiste simplement à obtenir les 50 derniers échantillons et à former le réseau.

3.3.2 Les méthodes d'optimisation utilisées

A- Optimisation par recherche

Les hyperparamètres sont importants pour les algorithmes d'apprentissage automatique car ils contrôlent directement les comportements des algorithmes d'apprentissage et ont un effet significatif sur les performances des modèles d'apprentissage automatique. Plusieurs techniques ont été développées et appliquées avec succès pour certains domaines d'application. Cependant, ce travail exige des connaissances professionnelles et l'expérience d'experts. Et parfois, il faut recourir à la recherche par la force brute. Par conséquent, si un algorithme efficace d'optimisation des hyperparamètres peut être développé pour optimiser une méthode d'apprentissage automatique donnée, cela améliorera considérablement l'efficacité de l'apprentissage automatique. Dans ce travail, nous envisageons de construire la relation entre la performance des modèles d'apprentissage automatique et leurs hyperparamètres par la méthode de recherche .

Pour cette raison nous nous sommes intéressés par les hypothèses suivantes pour chercher la combinaison la plus pertinente pour améliorer les résultats initiale du modèle CNN proposé : Dropout, Learning rate, Batch-Size, et le nombre d'epoch.

B- Grid search

Après avoir trouvé le résultat de la première expérience, nous avons utilisé la taille d'entraînement optimale de l'ensemble de données ainsi que la méthode de réglage des hyperparamètres pour découvrir l'impact du réglage des paramètres sur l'amélioration de la précision du modèle pour la reconnaissance des émotions. Nous avons utilisé la méthode de grille de recherche «Grid search», c'est la méthode la plus courante compte tenu de sa procédure simple et directe. Il s'agit d'une méthode de recherche non informée , ce qui signifie qu'elle n'apprend pas de ses itérations précédentes. Nous avons positionné une liste de possibilité pour chaque paramètre, puis l'utilisation de cette méthode implique de tester chaque combinaison unique d'hyperparamètres dans l'espace de recherche pour déterminer la combinaison qui donne les meilleures performances. Nous avons commencé par définir un dictionnaire pour la grille qui sera une entrée pour GridSeachCv. La figure suivante présente la grille d'espace de recherche utilisée dans notre modèle CNN proposée.

```
param_grid:
learnRate = [1e-3,1e-2,1e-1]
dropout = [0.2,0.3]
batchSize = [4, 8, 16, 32]
epochs = [10, 20, 30, 40,50]
```

FIGURE III.13 – *Grid search de modéle CNN proposé*

Nous utiliserons GridSearchCV la classe de la bibliothèque Scikit-Learn pour cette optimisation. La première chose à mentionner est que la recherche par grille devra exécuter et comparer 120 modèles (=3*2*4*5, multiplication du nombre de valeurs sélectionnées).

4 Conclusion

Dans ce chapitre, nous avons testé l'apprentissage de transfert avec le modèle VGG-16 et VGG-19, qui a rencontré un grand succès avec l'essor du Deep Learning. En effet, les modèles utilisés dans le domaine de la reconnaissance des expressions faciales nécessitent des temps de calcul élevés et des ressources importantes. Cependant, en utilisant des modèles pré-entraînés comme point de départ, le Transfer Learning permet de développer rapidement des modèles performants et de résoudre efficacement des problèmes complexes. Après l'entraînement des modèles, nous constatons que les résultats obtenus sur l'ensemble de test en termes de précision et de perte pour le VGG-19 et le VGG-16 sont mauvais par rapport aux résultats de l'ensemble d'apprentissage. Nous avons donc poursuivi le développement d'un modèle CNN dans keras. Dans le chapitre suivant nous présenterons les résultats des réseaux de neurones convolutifs utilisés ainsi que la méthodologie d'apprentissage pour chaque modèle.

Chapitre IV

Expériences et évaluation

Sommaire

1 Introduction

Nous avons évoqué, dans la section précédente, la méthodologie proposée pour développer un modèle de reconnaissance des émotions faciales. Cette méthodologie consiste à tester le transfert learning pour la classification des images ainsi que le développement et l'optimisation d'un modèle cnn dans scratch.

Nous allons dans ce qui suit présenter les étapes de l'implémentation de notre approche et les différentes étapes de réalisation. Nous commençons tout d'abord par la présentation des ressources, du langage et de l'environnement de développement que nous avons utilisé, puis les étapes de la réalisation de notre modèle. Nous terminerons ce chapitre par la présentation des différents résultats expérimentaux obtenus, appuyés par des tableaux comparatifs, des courbes de visualisation des résultats ainsi que des matrices de confusion et une discussion détaillée.

2 Environnement de travail

Dans cette section, nous présenterons les environnements matériel et logiciel de notre travail.

2.1 Environnement matériel

Afin de mener à bien ce projet, il a été mis à notre disposition un ordinateur portable Lonovo avec les caractéristiques suivantes :

2.1.1 Environnement matériel

- Processeur : Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz.

- PRAM :8,00 Go (7,84 Go utilisable).

- Système d'exploitation : 64 bits, processeur x64, Windows 10 Professionnel.
- Carte graphique : Intel® UHD Graphics for 10th Gen Intel® Processors.

Nous avons aussi exécuter les modèles proposés sur Google colab pour profiter de ses GPU gratuits et optimiser le temps de calcul en local.

2.1.2 Environnement logiciel

Notre choix s'est porté sur Python comme un language de programmation, la bibliothèque de vision par ordinateur Opencv, Tensorflow de google et Keras comme un framework pour la mise en oeuvre de notre réseau de neurones, ainsi que d'autres bibliothèques (Numpy, Matplotlib,etc...), en plus du notebook Jupyter et google colab pour l'édition et test de notre code.

FIGURE IV.1 – *Le résultat de détection des points caractéristiques à partir du visage en utilisant dlib[22].*

2.2 Bibliothèques utilisées

— **Keras :** Elle a été initialement écrite par François Chollet c'est est une bibliothèque open source écrite en python et nous permet de manipuler les algorithmes de réseaux de neurones profonds et de machine Learning, notamment Tensorflow et Theano de créer de nouvelles couches, des fonctions et développer des modèles à la pointe de la technologie avec peu de restrictions en quelques lignes de code.

— **Tensorflow :** c'est un framework multi-platformes de programmation pour le calcul numérique de Google initié et développé par l'équipe Google Brain spécialisé dans l'intelligence artificielle, et rendu Open Source en Novembre 2015, et devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning, Supporte les calculs sur CPU, GPU et même le calcul distribue sur cluster, le Temps de compilation est très courts, Cependant dans FACECNN nous utilisant Cette plateforme via la bibliotheque Keras ce qui est sur-couche a TensorFlow donc un niveau plus haut que Tensorflow.

— **OpenCV (Open Source Computer Vision Library) :** c'est une bibliothèque qui propose un ensemble de plus de 3000 algorithmes de vision par ordinateur dans le domaine de traitement d'images, accessible au travers d'API pour les langages Python, C et C++.

— **Numpy :** Est une bibliothèque permet de réaliser des calculs numériques en se basant sur Python. Elle introduit une gestion simple des tableaux de , des fonctions de diffusion on peut aussi l'intégrer le code C / C ++ et Fortran.

— **Matplotlib :** Est une bibliothèque de traçage pour Python et son extension mathématique numérique NumPy . Il fournit une API orientée objet permettant d'incorporer des graphiques dans des applications à l'aide de kits d'outils d'interface graphique à usage général tels que Tkinter , Python , Qt ou GTK +.

3 Les Base De Données

Nous évaluons la méthode proposée en utilisant des images faciales issues de deux bases de données accessibles publiquement : CK+ , et FER2013. Les deux bases de données sont disponibles sur Kaggel.

3.1 La base de données ck+

L'ensemble de données Extended Cohn-Kanade (CK+) est composé de 981 images réparties en sept classes (peur, dégoût, triste, heureux, neutre, surprise, en colère). Les images ont une taille de 48 × 48 avec une palette de couleurs à échelle de gris, comme illustré au niveau de la figure suivante.

FIGURE IV.2 – *Exemple d'images de la base ck+*

Les variations des classes et les distributions des caractéristiques sont utiles dans la phase de fusion pour que d'autres classes obtiennent une bonne distribution et formalisent la quantité de variation des données. En règle générale, les images prises à partir d'images vidéo n'ont pas beaucoup varié et le nombre total d'éléments est négligeable par rapport aux autres ensembles de données.

Par rapport au FER-2013, les images sont en vue frontale avec un motif net pour l'expression faciale. cK+48 contenait moins de 1000 images et n'avait pas de division dans les ensembles de formation et de validation comme FER-2013. Nous l'utiliserons dans le cadre de l'ensemble d'apprentissage final pour chaque intégration d'ensemble de données dans nos expériences. La figure suivante représente la distribution graphique des différentes classes dans l'ensemble de données CK+48, où les barres rouges indiquent les valeurs inférieures à la moyenne et le vert indique les valeurs supérieures à la moyenne, où la moyenne est égale à 142.

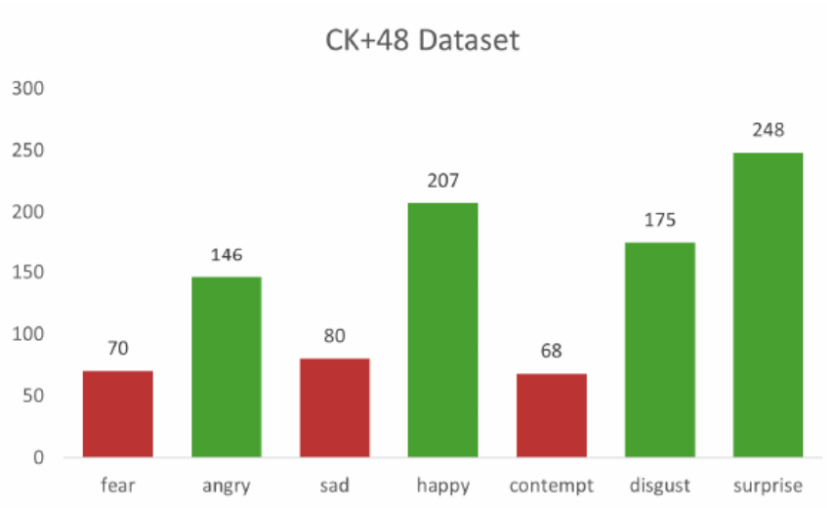


FIGURE IV.3 – *Répartition du jeu de données CK+48 avec la classification des expression*

3.2 La base de données FER2013

La base de données utilisée dans notre expérimentation est FER2013. Elle a été préparée par Pierre-Luc Carrier et Aaron Courville [FER]. Elle est disponible sur la plateforme des compétitions en science des données kaggle. La base d'image de FER2013 se compose de 35888 images de visages en niveaux de gris comme indiqué dans la figure suivante.

FIGURE IV.4 – *Exemple d'images de la base FER2013.*

Chaque image à une taille de 48x48, ces images sont étiquetées en 7 classes présenter dans le tableau 3.1 sous dessus. Dans cette base, on trouve 28 709 images pour l'apprentissage et 3 589 images pour le test. Nous choisissons cette base parce qu'elle contient des images en niveaux de gris éliminé les images en couleurs qui contient des données n'est pas nécessaires pour notre entraînement. Et une taille des images à une résolution (48 x 48) qui garantit que l'émotion est détectable [TKC05]. FER2013 nous épargnons de l'étape indispensable de pré-traitement des images souvent très coûteuse.

TABLE IV.1 – Répartition des images de la base FER2013 par classe.

Émotions	Nombres des images
Colère = 0	4593
Dégoût = 2	547
Peur = 3	5121
Heureux = 4	8989
Triste = 5	6077
Surprise = 6	4002
Neutre = 1	6198

La figure suivante représente la distribution graphique des différentes classes dans l'ensemble de données FER2013, où les barres rouges indiquent les valeurs inférieures à la moyenne et le vert indique les valeurs supérieures à la moyenne, où les moyennes sont égales à 4 100 et 1 100 pour les ensembles d'apprentissage et de validation.

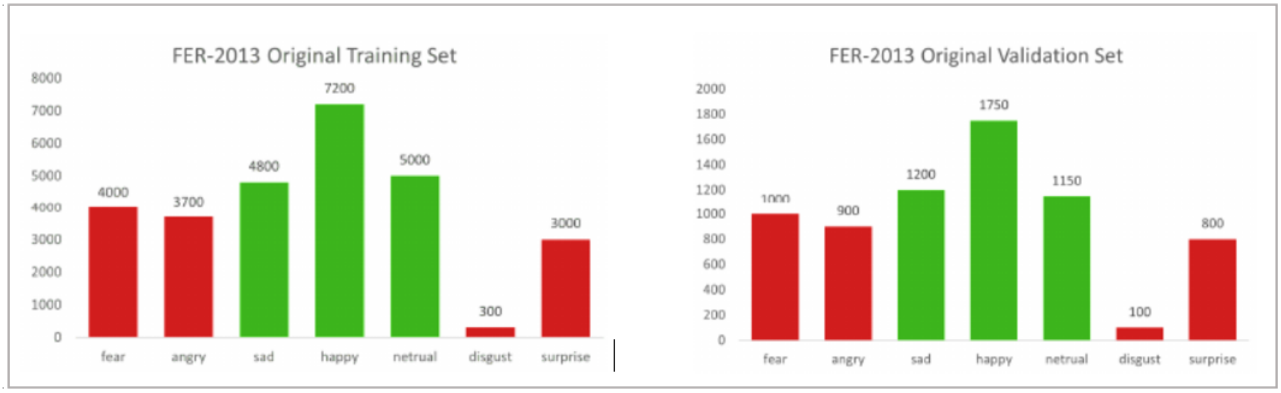


FIGURE IV.5 – *Répartition des classes dans le jeu de données FER-2013. Le graphique représente les images de validation et de test.*

4 Résultats et discussions

4.1 Première expérience : Apprentissage par transfert

Les données ont été collectées à partir des deux ensembles de données FER2013 et Ck+. Les échantillons de données de l'ensemble de données sont présentés dans la section précédente . Pour tester le modèle proposé, les étapes suivantes sont considérées pour arriver à la classification des images d'entrées par le transfert learning.

L'architecture utilisée est composée de deux étapes : Prétraitement et extraction et classification des caractéristiques. Ici, l'architecture proposée utilise la technique d'apprentissage par transfert avec le réseau vgg-16 pour extraire les caractéristiques et transmettre sa sortie à un classificateur softmax pour classer ces expressions.

Tout d'abord, la région d'intérêt (partie de visage) est extraite des images de visage données en utilisant l'algorithme Haarcascade mis en œuvre par OpenCV pour détecter le visage frontal. Ensuite, à partir des régions de visage recadrées, l'image a été redimensionnée en 224 × 224 pixels, nous avons inclus des poids égaux à 'imagenet' pour télécharger l'architecture réseau vgg-16 qui a été formée sur un énorme ensemble de données ImageNet et en même temps, nous avons également défini include top égal à 'False' pour ignorer les couches Fc du réseau préformé du téléchargement.

Après ce processus, les ensembles de validation et de test ont été créés respectivement à partir du modèle vgg-16 ImageNet. La forme d'entrée de la caractéristique de l'activation avant les couches entièrement connectés de chaque image qui vient après le passage de l'image à travers vgg-16 est de 7 × 7 × 512 avec une taille de lot égale à 64. Les vecteurs de caractéristiques ont été extraits de l'architecture de réseau vgg-16 proposée et les poids de tous les cinq blocs convolutionnels ont été congelés.

Une nouvelle couche dense a été introduite et ajoutée à droite après les dernières couches convolutives vgg-16 qui comprennent trois (3) couches 512, 256 et 128. La sortie du réseau est donnée à un nouveau classificateur et les images ont été classées dans l'une des septs expressions faciales de base en utilisant le fonction softmax. Le modèle proposé qui a été utilisé pour classer ces sept expressions faciales de base contiennent 13 couches de convolution de l'architecture de réseau vgg-16. La fonction d'activation qui a été utilisée dans les couches cachées est la fonction d'activation d'unité linéaire rectifiée (ReLu). Le réseau a été alimenté avec des images de taille 224 × 224 pixels comme images d'entrée par défaut. Il y a trois couches Fc de taille 128, 256 et 512 nœuds juste après les couches convolutionnelles vgg-16.

Comme la couche de convolution, la fonction d'activation relu a également été appliquée dans ces couches, puis après chaque couche cachée, nous avons inséré une couche de suppression et la valeur de nos couches de dropout est définie sur 0,25. Il désactive aléatoirement 25% des noeuds de la couche cachée pour éviter de sur apprentissage de réseau. À la dernière étape, la couche de sortie du réseau comprend sept nœuds car elle a 7 classes et la fonction d'activation qui a été utilisée dans la couche de sortie est le classificateur softmax. Adam a été utilisé comme optimiseur de notre modèle.

En raison des applications FER dans le monde en temps réel, il est devenu si célèbre et populaire ces dernières années, Ce travail de recherche propose un modèle pour la tâche de reconnaissance des expressions faciales qui utilise la technique d'apprentissage par transfert avec le modèle vgg-16. Pour prouver l'efficacité de notre modèle proposé, les deux ensemble de données CK+ et FER2013 ont été formés sur le modèle et les résultats montrent un taux de reconnaissance faible dans les sept classes. Afin de montrer les résultats obtenus pour notre modèle testé par le transfert Learning de modèle vgg-16 , on illustre dans ce qui suit le tableaux

des résultats en termes d'exactitude (accuracy) et d'erreur ainsi que les graphique pour la base de données CK+ et la base FER-2013.

TABLE IV.2 – Résultat de l'éxperience du transfert learning avec le modéle VGG16 testé sur la base CK+ et FER2013.

Modèle	Base de données	Loss	Accuracy	Val_loss	Val_accuracy	Précision
Modèle VGG-16	CK+	1.04	60.71%	1.23	56.80%	58%
	FEER-2013	0.79	69.63%	1.01	59.80%	64%

Pour l'évaluation des performances du modèle, nous adoptons à des graphiques de précision et de perte ainsi que la matrice de confusion comme indice d'évaluation.

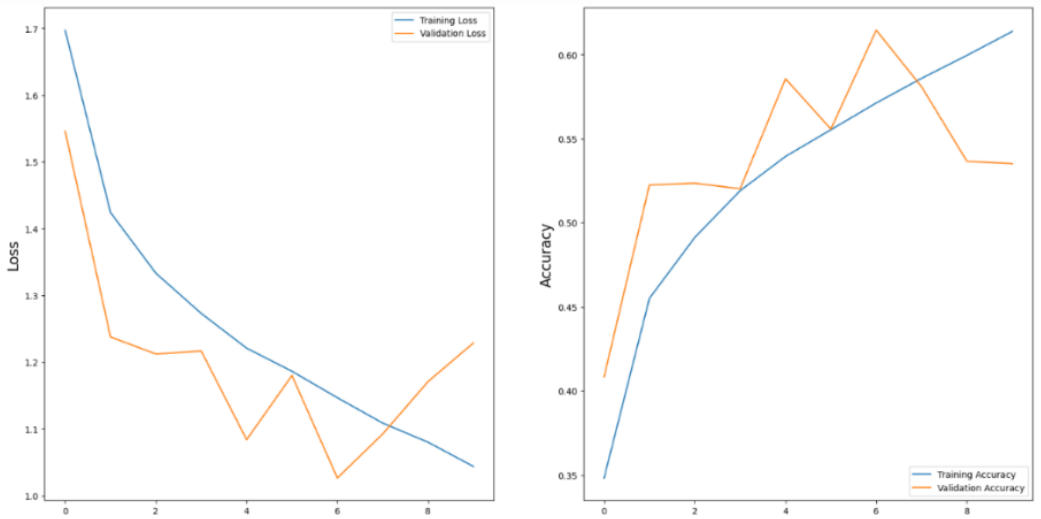


FIGURE IV.6 – *les métriques de performance sur l'ensemble de validation et de test de la base de données CK+*

Après l'entraînement de ce modèle sur l'ensemble de données CK+ nous constatons les remarques suivantes à partir des courbes d'accuracy et loss manifeste dans la figure précédente qui représentent le développement des résultats d'accuracy et de loss par rapport les époques (08 époques) que la précision du modèle sur l'ensemble d'entraînement et test sont respectivement de 60.71% et 56.80%.
Le nombre d'époques est le nombre de fois où le modèle parcourt les données. Plus nous avons d'époques, plus le modèle s'améliorera, jusqu'à un certain point. Puis, le modèle cessera de s'améliorer à chaque époque.
La perte s'est produite dans le processus de test égale à 1.04%, il indique l'erreur sur l'ensemble de données de test. Il s'agit du nombre total d'erreurs survenues pour l'ensemble de test lors du test du modèle après l'entraînement du modèle avec l'ensemble d'apprentissage.
Une matrice de confusion est en fait une sorte de disposition de tableau qui nous permet de calculer et nous permet également de visualiser les performances d'un algorithme. La matrice de confusion est également connue sous le nom de matrice

d'erreur et elle est principalement utilisée dans l'apprentissage automatique et plus particulièrement le problème de la classification statistique. Chaque colonne représente les instances dans une classe originale ou réelle tandis que chaque ligne de la matrice représente les instances dans une classe prédite (ou vice versa).

Pour l'évaluation des performances du modèle, nous adoptons à des graphiques de précision et de perte ainsi que la matrice de confusion comme indice d'évaluation.

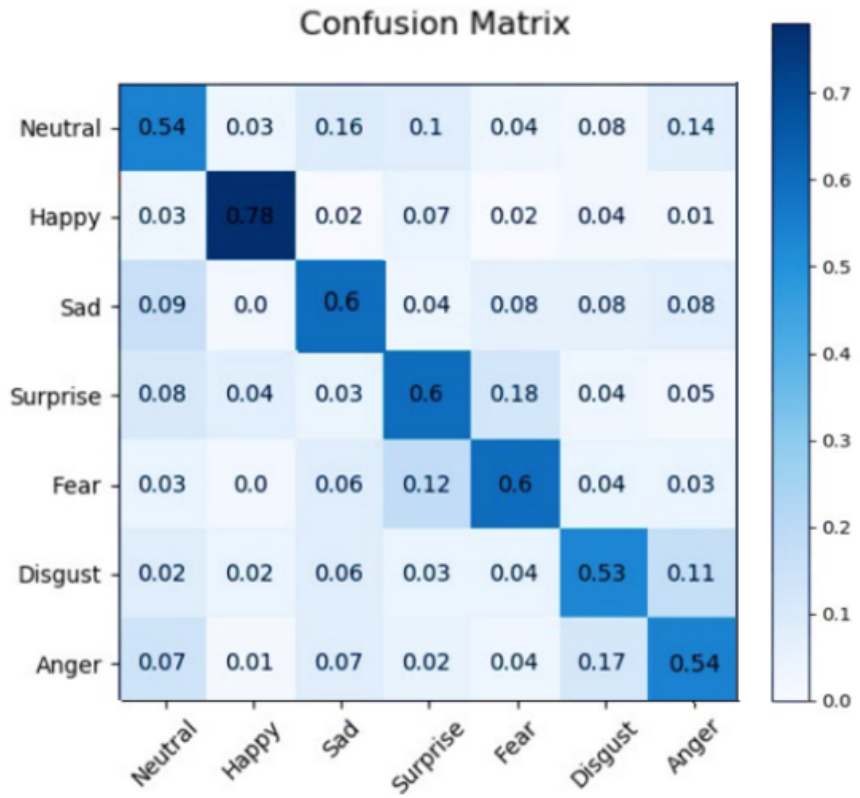


FIGURE IV.7 – *Matrice de confusion pour les sept classes expressions de la base CK+.*

La matrice de confusion de l'ensemble de données Ck+ est illustrée à la figure précédente, elle permet de bien comprendre les limites du cette base de données. Nous remarquons qu'il y a souvent confusion entre « Surprise», « peur » et « tristesse » parce qu'elles créent des mouvements semblables. Les matrices de confusion montrent également que "dégoût" est facilement mal classé comme "colère".

Notre modèle a été testé également sur la base FER2013, Dans ce qui est suit nous allons détaillées l'évaluation des performances du modèle par les graphique de précision et de perte ainsi que la matrice de confusion comme indice d'évaluation.

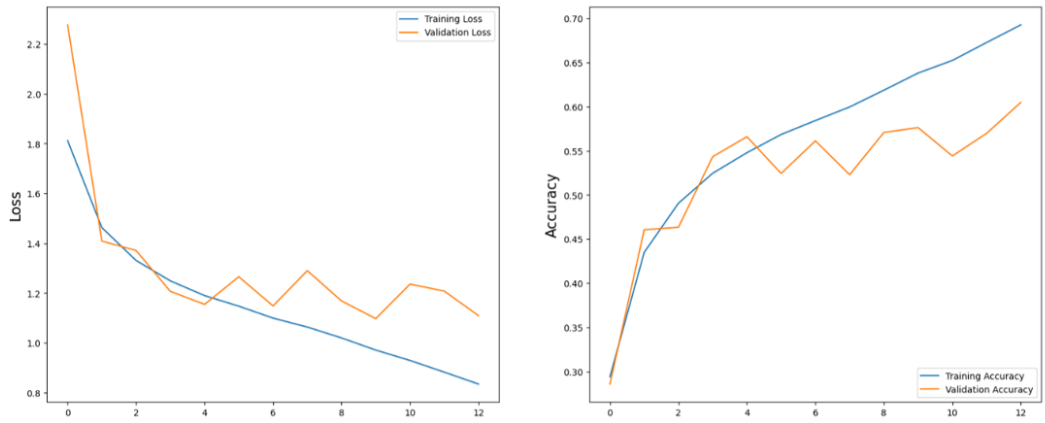


FIGURE IV.8 – *les métriques de performance sur l'ensemble de validation et de test de la base de données FER2013.*

La figure suivante représente la matrice de confusion qui permet d'évaluer la performance de notre modèle. La matrice illustre de près la classification des images dans chaque classe émotionnelle. A titre d'exemple, le modèle a bien classé les images heureux et il a mal classé les images de peur de la base FER2013.

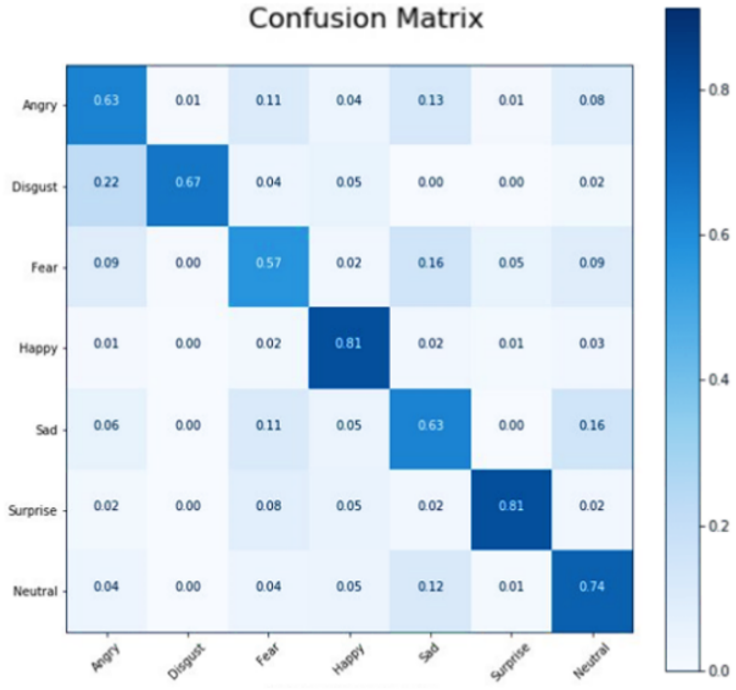


FIGURE IV.9 – *Matrice de confusion pour les sept classes expressions de la base FER2013.*

Nous avons testé notre modèle d'apprentissage par transfert VGG16 avec la base de données FER2013. L'accuracy de l'ensemble de données FER2013 était de 69.63%, la Val-accuracy était de 59.80%, et le perte était de 0.79% et 1.01% sur l'ensemble de test et de validation. Cette experience sur la FER-2013 présente une performance de reconnaissance d'expression faciale plus élevée que l'entraînement de

modèle sur la base Ck+.

On peut conclure que la précision de la plupart des expressions est moyennement présentée. Comme l'époque augmente de manière cohérente au cours de chaque cycle d'entraînement, le modèle sera bien adapté et fonctionnera bien pendant la période de test en temps réel. Les chiffres colorés de chaque classe indiquent que les données de recherche ont été bien catégorisées. De plus, les chiffres de chaque côté de la diagonale indiquent le pourcentage d'images qui ont été répertoriées de manière inappropriée. Comme ces nombres sont plus petits que les nombres sur la diagonale, on peut en déduire que l'algorithme a fonctionné correctement mais avec un taux de reconnaissance moyen.
les bases CK+ et FER2013 les deux ont une précision de 58% et 64%, respectivement, comparables l'une à l'autre. À la suite des résultats de la revue de la littérature, nous avons mené une analyse comparative, dans laquelle nous avons utilisé l'apprentissage par transfert pour évaluer les sentiments d'image. Le tableau suivant contient une collection d'études et de comparaisons de performances de modèles provenant de diverses sources testés sur les deux jeux de données utilisées dans notre étude.

TABLE IV.3 – Bilan comparatif des résultats trouvés avec les modèles existants entrainées sur CK+ et FER2013.

Base de données	**Méthode**	**Précision (%)**
FER2013	Wang [t22]	71.1%
	Pramerdorfer et al. [t46]	75.21%
	Kim et al. [t45]	73.37%
	Guo et al. [t44]	71.33%
	Shao et al. [t34]	71.14%
	Méthode proposé basé sur le VGG-16	**64%**
CK+	VGG Net + LSTM [t37]	97.2%
	Yang et al. [t39]	97.3%
	Zhang et al. [t41]	98.9%
	Turan [t24]	96.1%
	Shao et al. [t34]	95.29%
	Méthode proposé basé sur le VGG-16	**58%**

Comme le montrent ces résultats, le modèle proposée s'est avéré avoir une précision inférieure que d'autres méthodes récentes. Dans ce cas, nous avons décidé de développer et optimiser les résultats initiales d' un modèle de classification des expressions facile à zéro dans keras. Dans ce qui suit, nous présentons par détail la deuxième expérience réalisée.

4.2 Deuxième expérience : Optimisation de modèle CNN Proposé

Pour tester les performances du cadre présenté par détails au niveau de chapitre précédent, nous avons testé le modèle sur la base CK+. Les résultats initiaux de

modèle CNN proposé sont les suivantes :

TABLE IV.4 – Résultat initiale de modèle CNN proposé testé sur la base CK+

Modèle	Base de données	Loss	Accuracy	Val_loss	Val_accuracy	Précissio
Modèle CNN proposé	CK+	0.3200	91.97%	0.5435	95.93%	93.9%

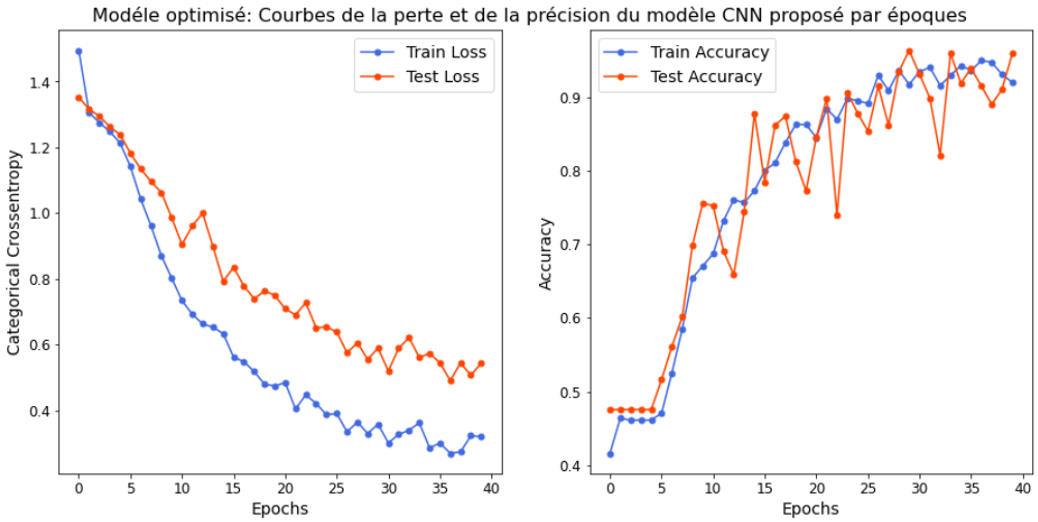


FIGURE IV.10 – *Résultat de perfermence initiale du modéle CNN from scratch .*

Nous avons obtenu une précision de près de 94% sur le jeu de données CK+ en exécutant 40 époques.

4.2.1 Optimisation du modéle par la méthode de recherche des hyperparamétre

Le but de cette expérience est d'étudier l'efficacité de la méthode REF proposée avec le jeu de données CK+.
Les travaux de Bellamkonda [Bel21] a mené des recherches sur la reconnaissance des émotions faciales (REF) par CNN hyperparamétrique en utilisant GA. Le jeu de données utilisé pour la recherche est FER2013, pour reconnaître sept émotions : peur, joie, surprise, tristesse, neutre, dégoût et colère. L'étude expérimentale se concentre sur l'impact des méthodes GA dans l'ajustement des hyperparamètres. Chalavadi et al. [IC16] ont fait des recherches sur la reconnaissance des émotions humaines en utilisant CNN et GA. Un algorithme de descente de gradient est utilisé pour entraîner les classificateurs CNN (pour trouver un minimum local) lors de l'évaluation de la pertinence des chromosomes de l'AG. Le site capacités de recherche globale de GA et la capacité de recherche locale de l'algorithme de descente de gradient sont exploitées pour trouver une solution plus proche de l'optimum global. Ils ont montré que la combinaison des preuves du classificateur générées à l'aide d'algorithmes génétiques pour améliorer leurs recherches ont été menées sur le jeu de données UCF50.

Boughida et al. [BKL22] ont effectué leurs recherches en développant un modèle de reconnaissance des émotions humaines basées sur les filtres de Gabor et GA. Les caractéristiques de Gabor extraites les régions d'intérêt du visage humain détectées à l'aide de marqueurs faciaux. En outre, un algorithme génétique a été conçu pour optimiser les hyperparamètres de la machine à vecteurs de support (SVM) et sélectionner simultanément les meilleures caractéristiques. Des recherches ont été menées sur les jeux JAFFE et CK+ respectivement.
Grigorasi et Grigore [BC22] ont effectué leur recherche sur l'optimisation des hyperparamètres d'un CNN pour augmenter la précision dans le contexte de reconnaissance des expression faciale utilisant un algorithme de recherche appliqué à un espace de recherche défini par des valeurs discrètes d'hyperparamètres. L'ensemble de données utilisé pour la recherche est FER2013, pour reconnaître sept émotions (peur, heureux, surprise, triste, neutre, dégoût et colère) La recherche porte sur l'étude de l'impact de l'algorithme de recherche aléatoire pour le réglage des hyperparamètres sur le modèle CNN.
Pane et al. [PWP19] ont effectué leurs recherches pour améliorer la précision en utilisant le classificateur Random Forest ainsi que l'optimisation de la recherche de grille pour ajuster les paramètres. Les émotions reconnues sont la joie, la colère, la tristesse et la détente. Le jeu de données utilisé pour cette recherche est le jeu de données DEAP.
D'après les articles mentionnés ci-dessus, des algorithmes de réglage des paramètres tels que GA, Random Search, Grid Search ont été utilisés pour améliorer la précision des performances des modèles DL. Une revue de la littérature est effectuée pour étudier les différents algorithmes de réglage des paramètres. Le tableau suivant présente les résultats de l'expérience de recherche des hyperparamètres pour améliorer le résultat initial trouvé.
Please add the following required packages to your document preamble :

TABLE IV.5 – Résultats des expériences d'optimisation des hyperparamètres par la méthode de recherche.

La base de données	Nombre d'epoch	Learning Rate	Dropout	Accurarcy	Loss
CK+	50	0.01	0.15	66.58%	2.199
	50	0.01	0.3	86.56%	0.3083
	50	0.001	0.5	72.54.23%	0.4157
	50	0.01	0.2	87.80%	0.5184
	40	0.001	0.2	97.22%	0.1224
	40	0.001	0.5	82.50%	0.4238
	40	0.001	0.1	95.01%	0.1222

Il a été prouvé que l'approche de réglage des hyperparamètres par recherche est trés lente. Elle nécessite beaucoup de temps pour exécuter à chaque fois le modèle. Tandis que l'algorithme de réglage des paramètres de recherche par grille est un peu plus rapide. Alors, nous avons choisi d'effectuer une optimisation par grid search à fin d'améliorer la précision de modèle. Le modèle est entraîné et validé

pour trouver la meilleure précision et les résultats obtenus ont été analysés.

4.2.2 Optimisation du modèle par la méthode de Grid search

Nous avons étudié les différentes métriques de performance du modèle CNN après avoir effectué un réglage des hyperparamètres avec grid search. Le modèle a été entraîné et testé plusieurs fois pour obtenir des résultats précis. Le réglage des paramètres a permis non seulement d'améliorer la précision du modèle, mais aussi de réduire son temps d'exécution. L'expérience a produit les résultats suivants, comme le montrent la figure suivante.

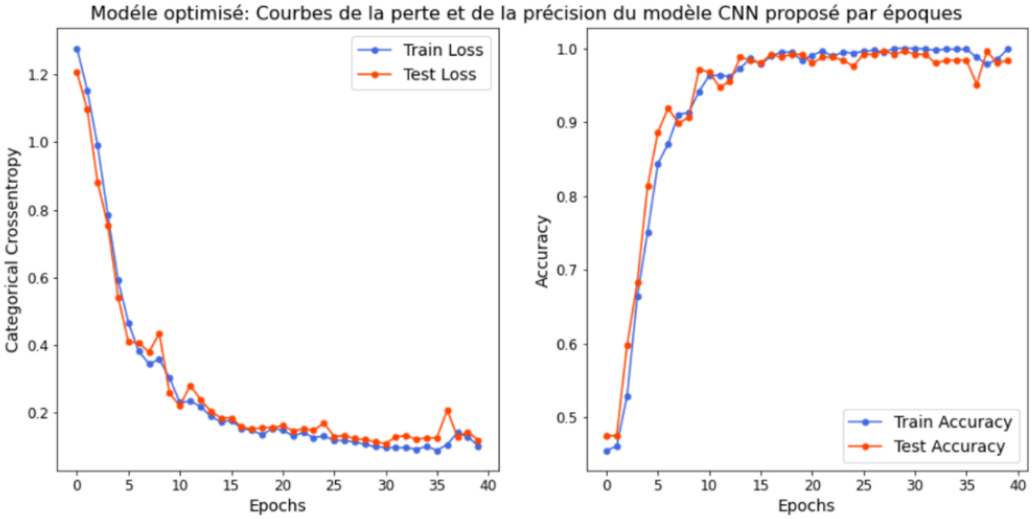


FIGURE IV.11 – *Graphiques de précision et de perte du modèle optimisé par la méthode de grid search testé sur la base CK+.*

Les graphiques de la figure ci-dessus illustrent la précision et la perte du modèle de reconnaissance des émotions faciales proposé dans ce travail. L'axe des x représente le nombre d'époques et l'axe des y représente la précision du modèle. La ligne bleu indique la précision de l'ensemble de formation et la ligne rouge indique la précision de l'ensemble de test dans les graphiques de précision et de perte. GridSearchCV a pris 1841 secondes pour 120 réglages de paramètres candidats.
Le simple réglage de certains hyperparamètres a augmenté la précision initiale de l'ensemble d'entraînement de 95% à 98% en passant 1841 secondes au réglage des hyperparamètres. les hyper paramètres optimaux d'entraînement de l'ensemble de données CK+ sur le modèle CNN proposé afin de fournir la meilleure précision pour la reconnaissance des émotions faciales sont présentés dans le tableau suivant :

TABLE IV.6 – Le résume des hyperparamètre optimaux

Hyperparamètre	Valeur
Nombre d'EPOCH	40
Learning Rate	0.001
Dropout	0.3
Batch size	32

L'historique des époques a montré une augmentation progressive de la précision et a atteint 97.94% de précision sur l'ensemble d'entraînement et 98,86% sur l'ensemble de test.

TABLE IV.7 – Tableau comparatif des résultats de perte et de précision de modèle initial avec le modèle optimisé.

Modèle	Base de données	Loss	Accuracy	Val_loss	Val_accuracy
Modèle initiale	**CK+**	0.3200	91.97%	0.5435	95.93%
Modèle optimisé		0.1124	97.94%	0.0966	98,86%

Cette analyse qui utilise le jeu de données CK+ montre que le modèle suggéré est comparable par les modèles existants en termes de performances tel qu'il est montré dans le tableau comparatif précédent. Plusieurs hyper-paramètres à prendre en compte lors de la construction d'un modèle d'apprentissage en profondeur ou d'apprentissage par transfert, qui visent tous à améliorer l'apprentissage et à réduire les pertes. De nombreux hyperparamètres différents peuvent être utilisés pour régler le processus d'apprentissage, tels que le taux d'apprentissage, le nombre d'itérations, le dropout et le choix de la fonction d'activation.Le modèle proposé a subi de nombreuses itérations pour améliorer ses performances.

Le modèle de phase de formation peut apprendre efficacement, mais cela n'a pas d'impact sur les données de validation, ce qui entraîne une perte importante de données et un faible niveau de précision de validation. Par conséquent, nous avons répété la même procédure jusqu'à ce que nous trouvions de meilleures valeurs pour les hyper-paramètres du modèle.

La recherche de grille trouve toujours le modèle le plus performant avec les valeurs d'hyperparamètres mentionnées dans la grille. Il est également facile à mettre en œuvre et à expliquer. Cependant, avec le nombre croissant d'hyperparamètres et de valeurs à tester, il peut facilement devenir coûteux en calcul car il modélise toutes les combinaisons d'hyperparamètres. Elle joue un rôle extrêmement important. Mais si la grille de paramètres est mal sélectionnée, l'algorithme de recherche ne trouve pas la meilleure combinaison.

TABLE IV.8 – Comparaisons de performances des méthodes existantes testé sur l'ensemble de données CK+ avec notre méthode proposé.

Base de données	Méthode	Précision (%)
CK+	VGG Net + LSTM [t37]	97.2%
	Yang et al. [t39]	97.3%
	Zhang et al. [t41]	98.9%
	Turan [t24]	96.1%
	Shao et al. [t34]	95.29%
	Modèle CNN proposé optimisé	**98.8%**

5 Conclusion

Lorsque nous avons lancé ce projet, nous avions deux objectifs, à savoir atteindre une bonne précision et appliquer les modèles REF au monde réel. Nous avons exploré plusieurs modèles, notamment des CNN peu profonds et des réseaux préformés basés sur VGG16 et VGG19. Dans ce chapitre, nous avons aussi défini la démarche que nous avons suivie pour arriver à optimiser les résultats initiaux d'un modèle CNN. Nous avons également introduit les détails de notre implémentation ainsi que nous avons présenté les résultats obtenus et nous avons comparés la performance de notre méthode avec les méthodes existantes.

Conclusion générale

L'approche globale de l'analyse automatique des expressions faciales implique généralement trois étapes. Étant donné une image d'entrée ou une séquence d'images, la première étape consiste à localiser le visage, détecter un ensemble de points faciaux. Une fois le visage détecté, l'étape suivante consiste à extraire les caractéristiques du visage. L'étape finale prend comme entrée le vecteur de caractéristiques extrait précédemment pour effectuer l'étape de classification. La classification d'images est une tâche importante dans le domaine de la vision par ordinateur, la reconnaissance d'objets et l'apprentissage automatique.

A travers ce mémoire, nous avons fait une généralité sur la reconnaissance des expressions faciales et l'utilisation d'un réseau de neurones convolutif. En parcourant les différents chapitres, nous avons décrit et clarifié la définition de la reconnaissance des expressions faciales, l'architecture de CNN, les étapes de reconnaissance faciale et l'optimisation des hyperparamètres d'un modèle CNN.
Au début de ce rapport, les défis rencontrés par la communauté de la vision par ordinateur pour reconnaître automatiquement les expressions faciales sont mentionnés. Ces défis incluent la complexité des calculs en terme du temps, la précision, la grande variabilité des données, les contraintes applicatives et matérielles.

C'est dans ce cadre que s'inscrit notre travail, qui a pour objectif de proposer une application intelligente de reconnaissance des expressions faciales (peur, heureux, surprise, triste, neutre, dégoût et colère), nous avons utilisé deux pistes de modélisation différentes (From scratch et l'apprentissage par transfert). Nous avons testé les deux modèles VGG16 et VGG19 sur les bases CK+ et FER2013. En comparant les résultats obtenus avec l'état de l'art, nous avons constatés que les travaux précédente offre des modèles avec une précision plus importante. ce qui nous a amené à développer un modèle CNN dans keras qui répond exactement à nos besoin. Après l'optimisation du modèle proposée nous avons obtenus 98.8% comme valeur de précision sur la base ck+.

Comme perspectives, nous envisageons différentes pistes pour renforcer l'efficacité. nous souhaitons porter ce travail sur des environnements haut de calcul : GPU, Clowd, Clusters. et explorer les avantages des facteurs d'architectures de paramètres, hyperparamètres permettant d'améliorer la qualité du réseau., appliquer notre modèle sur des images 3D acquises par des cameras de profondeur et Tester notre modèle sur d'autres bases des données 3D tel que BU-3DFE , et aussi inclure plus de classes, donc étendre le REF pour reconnaitre les micro-expressions, ce qui permettra plus de précision pour indiquer l'état émotionnel.

En conclusion, ce projet nous a permis d'acquérir de nouvelles connaissances. Nous avons pu découvrir au cours de ce travail, de nouvelles notions tel que la notion de l'émotion et la manipulation de l'expression faciale et de faire connaissance

des difficultés de confondre entre l'émotion et l'expression faciale et particulièrement la découverte de l'interaction Homme-Machine durant la reconnaissance des expressions faciales.

Bibliographie

[AAV00] Igor Aizenberg, Naum N Aizenberg, and Joos PL Vandewalle. *Multi-valued and universal binary neurons : Theory, learning and applications*. Springer Science & Business Media, 2000. vi, 22, 23, 25, 36

[AGAE14] Muzammil Abdulrahman, Tajuddeen R Gwadabe, Fahad J Abdu, and Alaa Eleyan. Gabor wavelet transform based facial expression recognition using pca and lbp. In *2014 22nd signal processing and communications applications conference (SIU)*, pages 2265–2268. IEEE, 2014. 29

[AHM+18] Sadia Arshid, Ayyaz Hussain, Asim Munir, Anum Nawaz, and Sanneya Aziz. Multi-stage binary patterns for facial expression recognition in real world. *Cluster Computing*, 21(1) :323–331, 2018. 30

[AK06] Petar S Aleksic and Aggelos K Katsaggelos. Automatic facial expression recognition using facial animation parameters and multistream hmms. *IEEE Transactions on Information Forensics and Security*, 1(1) :3–11, 2006. 17

[Alv13] Nelson Torro Alves. Recognition of static and dynamic facial expressions : a study review. *Estudos de Psicologia (Natal)*, 18 :125–130, 2013. 13

[ASC05] Zara Ambadar, Jonathan W Schooler, and Jeffrey F Cohn. Deciphering the enigmatic face : The importance of facial dynamics in interpreting subtle facial expressions. *Psychological science*, 16(5) :403–410, 2005. 13, 21

[Bas79] John N Bassili. Emotion recognition : the role of facial movement and the relative importance of upper and lower areas of the face. *Journal of personality and social psychology*, 37(11) :2049, 1979. 13

[BBADDB11] Stefano Berretti, Boulbaba Ben Amor, Mohamed Daoudi, and Alberto Del Bimbo. 3d facial expression recognition using sift descriptors of automatically detected keypoints. *The Visual Computer*, 27(11) :1021–1036, 2011. 31

[BC22] Lokesh Bejjagam and Reshmi Chakradhara. Facial emotion recognition using convolutional neural network with multiclass classification and bayesian optimization for hyper parameter tuning., 2022. 81

[Bel21] Satyachandra Saurabh Bellamkonda. Facial emotion recognition by hyper-parameter tuning of convolutional neural network using genetic algorithm, 2021. 80

[Bet12] Vinay Bettadapura. Face expression recognition and analysis : the state of the art. *arXiv preprint arXiv :1203.6722*, 2012. 17

[BH00] Imad A Basheer and Maha Hajmeer. Artificial neural networks : fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1) :3–31, 2000. 39

[BK14] Young-Hyen Byeon and Keun-Chang Kwak. Facial expression recognition using 3d convolutional neural network. *International journal of advanced computer science and applications*, 5(12), 2014. 19

[BKL22] Adil Boughida, Mohamed Nadjib Kouahla, and Yacine Lafifi. A novel approach for facial expression recognition based on gabor filters and genetic algorithm. *Evolving Systems*, 13(2) :331–345, 2022. 81

[BL97] Léandre Bouffard and Sylvie Lapierre. La mesure du bonheur. *Revue québécoise de psychologie*, 18(2) :271–310, 1997. 7

[BP+03] Ioan Buciu, IJCA Pitas, et al. Ica and gabor representation for facial expression recognition. In *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, volume 2, pages II–855. IEEE, 2003. 29

[BTA+15] Peter Burkert, Felix Trier, Muhammad Zeshan Afzal, Andreas Dengel, and Marcus Liwicki. Dexpression : Deep convolutional neural network for expression recognition. *arXiv preprint arXiv :1509.05371*, 2015. 16, 19

[Bul01] Peter Bull. Nonverbal communication. *The Psychologist*, 14 :644–647, 2001. 13, 14

[BY97] Michael J Black and Yaser Yacoob. Recognizing facial expressions in image sequences using local parameterized models of image motion. *International Journal of Computer Vision*, 25(1) :23–48, 1997. 13

[CDCLD15] Pierluigi Carcagnì, Marco Del Coco, Marco Leo, and Cosimo Distante. Facial expression recognition and histograms of oriented gradients : a comprehensive study. *SpringerPlus*, 4(1) :1–25, 2015. 29

[CHE] HADJER CHETTOUH. Montage d'un système de reconnaissance des expressions faciales avec le deep learning. vi, 8, 36

[CHFT06] Ya Chang, Changbo Hu, Rogerio Feris, and Matthew Turk. Manifold based analysis of facial expression. *Image and Vision Computing*, 24(6) :605–614, 2006. 28

[CKM+09] Jeffrey F Cohn, Tomas Simon Kruez, Iain Matthews, Ying Yang, Minh Hoai Nguyen, Margara Tejera Padilla, Feng Zhou, and Fernando De la Torre. Detecting depression from facial actions and vocal prosody. In *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–7. IEEE, 2009. 9, 10, 31

[CMK+06] George Caridakis, Lori Malatesta, Loic Kessous, Noam Amir, Amaryllis Raouzaiou, and Kostas Karpouzis. Modeling naturalistic affective states via facial and vocal expressions recognition. In *Proceedings of the 8th international conference on Multimodal interfaces*, pages 146–154, 2006. 31

[Col03] Jean-Marc Colletta. *Les émotions : cognition, langage et développement*, volume 247. Editions Mardaga, 2003. 24

[CSG+03] Ira Cohen, Nicu Sebe, Ashutosh Garg, Lawrence S Chen, and Thomas S Huang. Facial expression recognition from video sequences : temporal and static modeling. *Computer Vision and image understanding*, 91(1-2) :160–187, 2003. 27

[Dar65] Charles Darwin. 1872. the expression of emotions in man and animals, 1965. 6

[DBD15] Arnaud Dapogny, Kevin Bailly, and Séverine Dubuisson. Dynamic facial expression recognition by joint static and multi-time gap transition classification. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 1, pages 1–6. IEEE, 2015. 2, 31

[DBH+99] Gianluca Donato, Marian Stewart Bartlett, Joseph C. Hager, Paul Ekman, and Terrence J. Sejnowski. Classifying facial actions. *IEEE Transactions on pattern analysis and machine intelligence*, 21(10) :974–989, 1999. 7

[DGL13] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013. 16

[DJZ+05] Hong-Bo Deng, Lian-Wen Jin, Li-Xin Zhen, Jian-Cheng Huang, et al. A new facial expression recognition method based on local gabor filter bank and pca plus lda. *International Journal of Information Technology*, 11(11) :86–96, 2005. 29

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 29

[DTP12] Antonios Danelakis, Theoharis Theoharis, and Ioannis Pratikakis. 3d mesh video retrieval : A survey. In *2012 3DTV-Conference : The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4. IEEE, 2012. 25

[Eck03] Paul Eckman. Emotions revealed. *St. Martin's Griffin, New York*, 2003. 7

[EF71] Paul Ekman and Wallace V Friesen. Constants across cultures in the face and emotion. *Journal of personality and social psychology*, 17(2) :124, 1971. 8

[EF03] Paul Ekman and Wallace V Friesen. *Unmasking the face : A guide to recognizing emotions from facial clues*, volume 10. Ishk, 2003. 7

[EFA80] Paul Ekman, Wallace V Freisen, and Sonia Ancoli. Facial signs of emotional experience. *Journal of personality and social psychology*, 39(6) :1125, 1980. 7, 8

[Ekm89] Paul Ekman. The argument and evidence about universals in facial expressions. *Handbook of social psychophysiology*, 143 :164, 1989. 6

[Ekm03] Paul Ekman. Darwin, deception, and facial expression. *Annals of the new York Academy of sciences*, 1000(1) :205–221, 2003. 1, 2, 12

[EO91] Paul Ekman and Maureen O'Sullivan. Who can catch a liar ? *American psychologist*, 46(9) :913, 1991. 12

[EVM17] Joy Egede, Michel Valstar, and Brais Martinez. Fusing deep learned and hand-crafted features of appearance, shape, and dynamics for automatic pain estimation. In *2017 12th IEEE international conference on automatic face & gesture recognition (FG 2017)*, pages 689–696. IEEE, 2017. 10

[FDWS91] Jose-Miguel Fernández-Dols, Harald Wallbott, and Flor Sanchez. Emotion category accessibility and the decoding of emotion from facial expression and context. *Journal of Nonverbal Behavior*, 15(2) :107–123, 1991. 9, 23, 24

[FEF93] Mark G Frank, Paul Ekman, and Wallace V Friesen. Behavioral markers and recognizability of the smile of enjoyment. *Journal of personality and social psychology*, 64(1) :83, 1993. 12

[FGMR10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9) :1627–1645, 2010. 24

[FH05] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International journal of computer vision*, 61(1) :55–79, 2005. 24

[FL03] Beat Fasel and Juergen Luettin. Automatic facial expression analysis : a survey. *Pattern recognition*, 36(1) :259–275, 2003. 2, 6

[FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1) :119–139, 1997. 2, 24

[FT05] Nickolaos Fragopanagos and John G Taylor. Emotion recognition in human–computer interaction. *Neural Networks*, 18(4) :389–405, 2005. 31

[FZO+11] Tianhong Fang, Xi Zhao, Omar Ocegueda, Shishir K Shah, and Ioannis A Kakadiaris. 3d facial expression recognition : A perspective on promises and challenges. In *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, pages 603–610. IEEE, 2011. 13

[Gha10] Khadoudja Ghanem. Reconnaissance des expressions faciales à base d'informations video ; estimation de l'intensité des expressions faciales. 2010. vi, 15

[Gha16] Sonia Gharsalli. *Reconnaissance des émotions par traitement d'images*. PhD thesis, Université d'Orléans, 2016. 22

[HALL07] Chang Huang, Haizhou Ai, Yuan Li, and Shihong Lao. High-performance rotation invariant multiview face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 29(4) :671–686, 2007. 25

[HM17] Behzad Hasani and Mohammad H Mahoor. Facial expression recognition using enhanced deep 3d convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 30–40, 2017. 19

[HOH+07] Benjamin Hernandez, Gustavo Olague, Riad Hammoud, Leonardo Trujillo, and Eva Romero. Visual learning of texture descriptors for facial expression recognition in thermal imagery. *Computer Vision and Image Understanding*, 106(2-3) :258–269, 2007. 31

[HOLJ14] Hu Han, Charles Otto, Xiaoming Liu, and Anil K Jain. Demographic estimation from face images : Human vs. machine performance. *IEEE transactions on pattern analysis and machine intelligence*, 37(6) :1148–1161, 2014. 31

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7) :1527–1554, 2006. 26

[HSAC11] Di Huang, Caifeng Shan, Mohsen Ardebilian, and Liming Chen. Facial image analysis based on local binary patterns : A survey. *IEEE Transactions on Image Processing*, 2011. 29

[HT04] Kohsia S Huang and Mohan M Trivedi. Robust real-time detection, tracking, and pose estimation of faces in video streams. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 965–968. IEEE, 2004. 32

[HT17] Ross P Holder and Jules R Tapamo. Improved gradient local ternary patterns for facial expression recognition. *EURASIP Journal on Image and Video Processing*, 2017(1) :1–15, 2017. 30

[HWL+13] Shan He, Shangfei Wang, Wuwei Lan, Huan Fu, and Qiang Ji. Facial expression recognition using deep boltzmann machine from thermal infrared images. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 239–244. IEEE, 2013. 32

[IC16] Earnest Paul Ijjina and Krishna Mohan Chalavadi. Human action recognition using genetic algorithms and convolutional neural networks. *Pattern recognition*, 59 :199–212, 2016. 80

[JK18] Mira Jeong and Byoung Chul Ko. Driver's facial expression recognition in real-time for safe driving. *Sensors*, 18(12) :4270, 2018. 19

[JLY+15] Heechul Jung, Sihaeng Lee, Junho Yim, Sunjeong Park, and Junmo Kim. Joint fine-tuning in deep neural networks for facial expression

recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 2983–2991, 2015. 18

[JV03] Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3(14) :2, 2003. 25

[JZH20] Deepak Kumar Jain, Zhang Zhang, and Kaiqi Huang. Multi angle optimal pattern-based deep learning for automatic facial expression recognition. *Pattern Recognition Letters*, 139 :157–165, 2020. 19

[KBWK14] Volkan Kılıç, Mark Barnard, Wenwu Wang, and Josef Kittler. Audio assisted robust visual tracking with adaptive particle filtering. *IEEE Transactions on Multimedia*, 17(2) :186–200, 2014. 13

[KL16] Ali Mehmood Khan and Michael Lawo. Wearable recognition system for emotional states using physiological devices. *eTELEMED*, 2016 :131–137, 2016. 18

[Ko18] Byoung Chul Ko. A brief review of facial emotion recognition based on visual information. *sensors*, 18(2) :401, 2018. 16, 17

[KP06] Irene Kotsia and Ioannis Pitas. Facial expression recognition in image sequences using geometric deformation features and support vector machines. *IEEE transactions on image processing*, 16(1) :172–187, 2006. 27, 31

[KPB+13] Samira Ebrahimi Kahou, Christopher Pal, Xavier Bouthillier, Pierre Froumenty, Çaglar Gülçehre, Roland Memisevic, Pascal Vincent, Aaron Courville, Yoshua Bengio, Raul Chandias Ferrari, et al. Combining modality specific deep neural networks for emotion recognition in video. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 543–550, 2013. 31

[KQP03] Ashish Kapoor, Yuan Qi, and Rosalind W Picard. Fully automatic upper facial action recognition. In *2003 IEEE International SOI Conference. Proceedings (Cat. No. 03CH37443)*, pages 195–202. IEEE, 2003. 27

[KRP12] Sebastian Kaltwang, Ognjen Rudovic, and Maja Pantic. Continuous pain intensity estimation from facial expressions. In *International Symposium on Visual Computing*, pages 368–377. Springer, 2012. 28

[LBA99] Michael J Lyons, Julien Budynek, and Shigeru Akamatsu. Automatic classification of single facial images. *IEEE transactions on pattern analysis and machine intelligence*, 21(12) :1357–1362, 1999. 31

[LBL09] Gwen C Littlewort, Marian Stewart Bartlett, and Kang Lee. Automatic coding of facial expressions displayed during posed and genuine pain. *Image and Vision Computing*, 27(12) :1797–1803, 2009. 31

[LCP+11] Patrick Lucey, Jeffrey F Cohn, Kenneth M Prkachin, Patricia E Solomon, and Iain Matthews. Painful data : The unbc-mcmaster shoulder pain expression archive database. In *2011 IEEE International Conference on Automatic Face & Gesture Recognition (FG)*, pages 57–64. IEEE, 2011. 28

[LDY19] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 52(2) :1089–1106, 2019. 24

[LHMT14] Ping Liu, Shizhong Han, Zibo Meng, and Yan Tong. Facial expression recognition via a boosted deep belief network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1805–1812, 2014. 31

[LSWC14] Mengyi Liu, Shiguang Shan, Ruiping Wang, and Xilin Chen. Learning expressionlets on spatio-temporal manifold for dynamic facial expression recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1749–1756, 2014. 31

[LWL+14] Mengyi Liu, Ruiping Wang, Shaoxin Li, Shiguang Shan, Zhiwu Huang, and Xilin Chen. Combining multiple kernel methods on riemannian manifold for emotion recognition in the wild. In *Proceedings of the 16th International Conference on multimodal interaction*, pages 494–501, 2014. 27

[LZ04] Stan Z Li and ZhenQiu Zhang. Floatboost learning and statistical face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 26(9) :1112–1123, 2004. 25

[MB04] Iain Matthews and Simon Baker. Active appearance models revisited. *International journal of computer vision*, 60(2) :135–164, 2004. 28

[MD12] Aleix Martinez and Shichuan Du. A model of the perception of facial expressions of emotion by humans : research overview and perspectives. *Journal of Machine Learning Research*, 13(5), 2012. 30

[MW67] Albert Mehrabian and Morton Wiener. Decoding of inconsistent communications. *Journal of personality and social psychology*, 6(1) :109, 1967. 32

[NAC19] Foued NACER. Reconnaissance d'expression faciale à partir d'un visage réel. 2019. 7

[NRB+12] Jérémie Nicolle, Vincent Rapp, Kévin Bailly, Lionel Prevost, and Mohamed Chetouani. Robust continuous prediction of human emotions using multiscale dynamic cues. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 501–508, 2012. 31

[OPH96] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1) :51–59, 1996. 29

[PAHNM08] Axel Panning, Ayoub K Al-Hamadi, Robert Niese, and Bernd Michaelis. Facial expression recognition based on haar-like feature detection. *Pattern Recognition and Image Analysis*, 18(3) :447–452, 2008. 29

[PDRD96] P Jonathon Phillips, Sandor Z Der, Patrick J Rauss, and Or Z Der. *FERET (face recognition technology) recognition algorithm development and test results*. Army Research Laboratory Adelphi, MD, 1996. 23

[POP98] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 555–562. IEEE, 1998. 25, 32

[PP06] Maja Pantic and Ioannis Patras. Dynamics of facial expression : Recognition of facial actions and their temporal segments from face profile image sequences. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(2) :433–449, 2006. 27

[PR04] Maja Pantic and Leon JM Rothkrantz. Facial action recognition for facial expression analysis from static face images. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(3) :1449–1461, 2004. 7, 27

[PWP19] Evi Septiana Pane, Adhi Dharma Wibawa, and Mauridhi Hery Purnomo. Improving the accuracy of eeg emotion recognition by combining valence lateralization and ensemble learning with tuning parameters. *Cognitive processing*, 20(4) :405–417, 2019. 81

[RE20] Erika L Rosenberg and Paul Ekman. *What the face reveals : Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, 2020. 12

[RIM15] N Pattabhi Ramaiah, Earnest Paul Ijjina, and C Krishna Mohan. Illumination invariant face recognition using convolutional neural networks. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pages 1–4. IEEE, 2015. 27

[RRKC17] Byungyong Ryu, Adín Ramírez Rivera, Jaemyun Kim, and Oksam Chae. Local directional ternary pattern for facial expression recognition. *IEEE Transactions on Image Processing*, 26(12) :6006–6018, 2017. 30

[SBJJ17] Maryam Sultana, Muhammad Naeem Ali Bhatti, Sajid Javed, and Soon-Ki Jung. Local binary pattern variants-based adaptive texture features analysis for posed and nonposed facial expression recognition. *Journal of Electronic Imaging*, 26(5) :053017, 2017. 29

[SGM09] Caifeng Shan, Shaogang Gong, and Peter W McOwan. Facial expression recognition based on local binary patterns : A comprehensive study. *Image and vision Computing*, 27(6) :803–816, 2009. 29, 30

[Sha08] Caifeng Shan. *Inferring facial and body language*. PhD thesis, Queen Mary University of London, 2008. 13

[Shu07] ZHAO Shuji. Apprentissage et recherche par le contenu visuel de catégories sémantiques d'objets vidéo. 2007. 25

[SLS+07] Nicu Sebe, Michael S Lew, Yafei Sun, Ira Cohen, Theo Gevers, and Thomas S Huang. Authentic facial expression analysis. *Image and Vision Computing*, 25(12) :1856–1863, 2007. 31

[SN10] M Satiyan and R Nagarajan. Recognition of facial expression using haar-like feature extraction method. In *2010 International Conference on Intelligent and Advanced Systems*, pages 1–4. IEEE, 2010. 29

[SSG11] Albert Ali Salah, Nicu Sebe, and Theo Gevers. Communication and automatic interpretation of affect from facial expressions. In *Affective Computing and Interaction : Psychological, Cognitive and Neuroscientific Perspectives*, pages 157–183. IGI Global, 2011. 25

[SSYF20] Jamal Hussain Shah, Muhammad Sharif, Mussarat Yasmin, and Steven Lawrence Fernandes. Facial expressions classification and false label reduction using lda and threefold svm. *Pattern Recognition Letters*, 139 :166–173, 2020. 2, 29

[ST04] I-O Stathopoulou and George A Tsihrintzis. An improved neural-network-based face detection and facial expression classification system. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, volume 1, pages 666–671. IEEE, 2004. 7

[SYCG05] Shiguang Shan, Peng Yang, Xilin Chen, and Wen Gao. Adaboost gabor fisher classifier for face recognition. In *International Workshop on Analysis and Modeling of Faces and Gestures*, pages 279–292. Springer, 2005. 21

[SZ15] Reda Shbib and Shikun Zhou. Facial expression analysis using active shape model. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(1) :9–22, 2015. 28

[TC11] Fernando De la Torre and Jeffrey F Cohn. Facial expression analysis. *Visual analysis of humans*, pages 377–409, 2011. 32

[TKC01] Y-I Tian, Takeo Kanade, and Jeffrey F Cohn. Recognizing action units for facial expression analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 23(2) :97–115, 2001. 27

[TKC02] Ying-li Tian, Takeo Kanade, and Jeffrey F Cohn. Evaluation of gabor-wavelet-based facial action unit recognition in image sequences of increasing complexity. In *Proceedings of Fifth IEEE*

International Conference on Automatic Face Gesture Recognition, pages 229–234. IEEE, 2002. 29

[TKC05] Ying-Li Tian, Takeo Kanade, and Jeffrey F Cohn. Facial expression analysis. In *Handbook of face recognition*, pages 247–275. Springer, 2005. 12

[TNH11] Le Hoang Thai, Nguyen Do Thai Nguyen, and Tran Son Hai. A facial expression classification system integrating canny, principal component analysis and artificial neural network. *arXiv preprint arXiv :1111.4052*, 2011. 18

[TP91] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1) :71–86, 1991. 29

[TT10] Xiaoyang Tan and Bill Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE transactions on image processing*, 19(6) :1635–1650, 2010. 30

[VAG+15] Michel F Valstar, Timur Almaev, Jeffrey M Girard, Gary McKeown, Marc Mehu, Lijun Yin, Maja Pantic, and Jeffrey F Cohn. Fera 2015-second facial expression recognition and analysis challenge. In *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, volume 6, pages 1–8. IEEE, 2015. 28

[VGS+16] Michel Valstar, Jonathan Gratch, Björn Schuller, Fabien Ringeval, Denis Lalanne, Mercedes Torres Torres, Stefan Scherer, Giota Stratou, Roddy Cowie, and Maja Pantic. Avec 2016 : Depression, mood, and emotion recognition workshop and challenge. In *Proceedings of the 6th international workshop on audio/visual emotion challenge*, pages 3–10, 2016. 28

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001. 29

[VJ04] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2) :137–154, 2004. 24

[VP05] Danijela Vukadinovic and Maja Pantic. Fully automatic facial feature point detection using gabor feature based boosted classifiers. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1692–1698. IEEE, 2005. 27

[VP06] Michel Valstar and Maja Pantic. Fully automatic facial action unit detection and temporal analysis. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 149–149. IEEE, 2006. 27

[VP11] Michel F Valstar and Maja Pantic. Fully automatic recognition of the temporal phases of facial actions. *IEEE Transactions on*

Systems, Man, and Cybernetics, Part B (Cybernetics), 42(1) :28–43, 2011. 27

[VPAC06] Michel F Valstar, Maja Pantic, Zara Ambadar, and Jeffrey F Cohn. Spontaneous vs. posed facial behavior : automatic analysis of brow actions. In *Proceedings of the 8th international conference on Multimodal interfaces*, pages 162–170, 2006. 12

[XDlT13] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 532–539, 2013. 28

[XQR15] Xiaoming Xu, Changqin Quan, and Fuji Ren. Facial expression recognition based on gabor wavelet transform and histogram of oriented gradients. In *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 2117–2122. IEEE, 2015. 29

[YZFY04] Jian Yang, David Zhang, Alejandro F Frangi, and Jing-yu Yang. Two-dimensional pca : a new approach to appearance-based face representation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 26(1) :131–137, 2004. 17

[ZHT+11] Guoying Zhao, Xiaohua Huang, Matti Taini, Stan Z Li, and Matti PietikäInen. Facial expression recognition from near-infrared videos. *Image and vision computing*, 29(9) :607–619, 2011. 17

[ZJ05] Yongmian Zhang and Qiang Ji. Active and dynamic information fusion for facial expression understanding from image sequences. *IEEE Transactions on pattern analysis and machine intelligence*, 27(5) :699–714, 2005. 12

[ZLSA98] Zhengyou Zhang, Michael Lyons, Michael Schuster, and Shigeru Akamatsu. Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron. In *Proceedings Third IEEE International Conference on Automatic face and gesture recognition*, pages 454–459. IEEE, 1998. 27, 29

[ZP07] Guoying Zhao and Matti Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE transactions on pattern analysis and machine intelligence*, 29(6) :915–928, 2007. 29

[ZPRH07] Zhihong Zeng, Maja Pantic, Glenn I Roisman, and Thomas S Huang. A survey of affect recognition methods : audio, visual and spontaneous expressions. In *Proceedings of the 9th international conference on Multimodal interfaces*, pages 126–133, 2007. 30

[ZR12] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2879–2886. IEEE, 2012. 24

[ZZLQ16] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE signal processing letters*, 23(10) :1499–1503, 2016. 32