

AGENCE UNIVERSITAIRE DE LA FRANCOPHONIE (AUF)



INSTITUT FRANCOPHONE INTERNATIONAL (IFI)



MATIÈRE

RECONNAISSANCE DE FORME

RAPPORT FINAL

PROJET 1: RECONNAISSANCE DE VISAGES

Réalisé par :

KOBINA Pirizivè, Promo 20

MEDOU Daniel Magloire, Promo 21

NGASSA TCHOUDJEUH Tatiana, Promo 20

Enseignant:

Dr HO Tuong Vinh

Année Académique 2016 - 2017

INTRODUCTION

La reconnaissance dans un environnement contraignant, c'est-à-dire un environnement dans lequel un bon nombre de paramètres sont pris en compte, est tout à fait satisfaisante. Cependant, la reconnaissance du visage, dans un environnement incontrôlé, est encore un problème difficile en raison de certains problèmes techniques clés. De nombreux algorithmes et techniques ont été développés pour améliorer les performances de la reconnaissance de visage et de nos jours, le **deep learning** est très exploré pour les applications de la vision par ordinateur. Nous avons proposé l'implémentation avec le **Deep Convolutional Neural Network**.

I- DESCRIPTION DU PROBLÈME ET CARACTÉRISTIQUES UTILISÉES

1- Description du problème

Les visages humains sont des objets complexes avec des caractéristiques qui peuvent varier avec le temps. Cependant, nous, les humains avons la capacité naturelle de reconnaître les visages et d'identifier les personnes en un coup d'œil. Malheureusement, les ordinateurs ne disposent pas d'une telle capacité de reconnaissance. Ainsi, afin de les rendre autonomes dans le processus de reconnaissance des visages, de nombreux algorithmes ont été élaborés. Nous avons donc mis en place deux programmes de reconnaissance de visage dont un étant une solution basique avec la librairie OpenCV et l'autre basée sur le réseau neuronal convolutionnel.

2- Caractéristiques utilisées

Un visage étant de façon primaire reconnu par les caractéristiques suivantes: les yeux, le nez, la bouche et les oreilles. Souhaitant que notre programme ait les capacités humaines de reconnaissance de visage basées sur les caractéristiques précédemment cités. Le programme mis en place pourra extraire ces caractéristiques (face keypoints) pour faire la reconnaissance de visages.

II- IMPLEMENTATION

1- Outils

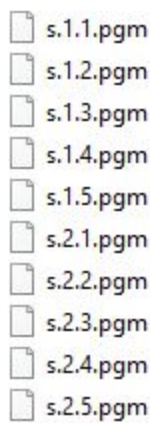
Dans le cadre de notre projet, les outils donc nous aurons besoin pour la réalisation de notre solution proposée seront entre autre le langage de programmation "Python" et la bibliothèque

multiplateforme “Opencv”. Pourquoi ces choix? La raison est simple car, Dans [L1], “Opencv” se concentre principalement sur le traitement d'image, la capture vidéo et l'analyse, y compris des fonctionnalités telles que la détection de visage et la détection d'objets, etc. Et dans notre projet nous allons manipuler les images d'où la nécessité de cette bibliothèque. En ce qui concerne “Python”, dans [L2], c’est un langage de programmation généralisé, interactif, orienté objet et de haut niveau et conçu pour être très lisible. Ce langage a plusieurs caractéristiques; est facile à apprendre, à lire, à maintenir, est extensible, portable, évolutif, etc. Comme **IDE**, nous allons utiliser **Spyder** qui est un environnement de développement interactif (**IDE**) gratuit inclu dans Anaconda. **Spyder** comprend des fonction d’édition, de test interactif, de débogage et d’introspection. Voilà en quelque sorte les raisons qui nous ont poussées au choix de ces deux outils pour la mise en place du programme.

2- Algorithme de l'implémentation

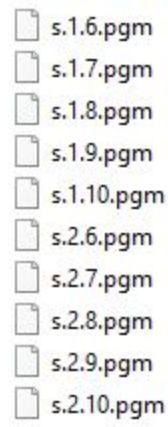
Afin de mieux expérimenter et de mieux évaluer notre solution, nous avons implémenté deux types solutions a savoir une solution simple basée sur la librairie OpenCV et une solution basée sur le CNN.

Pour toutes les solutions, nous disposons des méthodes nous permettant de séparer la base en deux c’est-à-dire les 5 premières images de chaque sujet formant la base d’apprentissage et les 5 dernières images de chaque sujet formant la base de test. Nous reconstituons les deux bases en nommant les images comme suit: “*s.numero.nom_image.extension*”. Chaque base est respectivement nommée *Train* et *Test*.



A vertical list of ten image file names: s.1.1.pgm, s.1.2.pgm, s.1.3.pgm, s.1.4.pgm, s.1.5.pgm, s.2.1.pgm, s.2.2.pgm, s.2.3.pgm, s.2.4.pgm, and s.2.5.pgm. Each name is preceded by a small square icon representing a file.

Fig 1: Train



A vertical list of ten image file names: s.1.6.pgm, s.1.7.pgm, s.1.8.pgm, s.1.9.pgm, s.1.10.pgm, s.2.6.pgm, s.2.7.pgm, s.2.8.pgm, s.2.9.pgm, and s.2.10.pgm. Each name is preceded by a small square icon representing a file.

Fig2: Test

a- Solution basique avec OpenCV

Pour cette solution nous avons utilisé la méthode de détection de visage *Haar-Like* implémentée dans OpenCV pour faire la détection de visages puis utiliser le *faceRecognizer*, toujours une méthode de OpenCV, pour faire la reconnaissance.

Nous créons, dans un premier temps, un modèle de reconnaissance de visage. Ensuite, nous passons à la phase d'apprentissage de différents visages présents dans le dossier *Train* avec le modèle créé puis nous sauvegardons ce modèle dans un dossier *recognizer*. A la phase de test, nous chargeons le modèle enregistré, puis, pour chaque image présent dans le dossier *Test* nous faisons une prédiction avec le modèle d'apprentissage chargé. La phase de prédiction permet d'attribuer une classe à l'image de test lue. Nous faisons une comparaison entre sa classe originale et celle prédite pour tirer conclusion sur le classement.

b- Solution avec CNN

Nous avons implémenté une solution basée sur le réseau neuronal convolutionnel. Dans cette solution, nous avons utilisé une architecture à 19 couches, ceci afin d'augmenter la performance d'apprentissage du modèle. Chaque couche permet de réduire les vecteurs de caractéristiques de chaque image. Dans [L3], la particularité du CNN est l'extraction automatique des caractéristiques et pour cela nous avons recompilé la librairie *Keras 2.0.2* pour la version 3.5 de Python. Après la construction des couches, nous passons à la phase d'apprentissage du modèle et à l'évaluation du modèle avec les données de test qui est illustrée par le taux de précision et le taux d'erreur.

3- Analyse des resultats

Nous avons construit chacun des deux modèles avec 50% des images de la base de données. Ensuite avec les 50% restants avons effectué une prédiction avec chacun des modèles construits. Pour chaque modèle, nous avons obtenu les résultats suivants:

```

** Construction de la base d'apprentissage **
** Construction de la base d'apprentissage terminee **
** Construction de la base de test **
** Construction de la base de test terminee **
train = 200 et test = 200
** Apprentissage en cours **
** Apprentissage termine **
s1 est correctement reconnu
s1 est mal reconnu comme etant s22
s1 est correctement reconnu
s1 est correctement reconnu
s10 est mal reconnu comme etant s1
s10 est mal reconnu comme etant s1
s10 est mal reconnu comme etant s19
s10 est mal reconnu comme etant s1
s10 est mal reconnu comme etant s33
s11 est mal reconnu comme etant s34
..
..
..
s5 est mal reconnu comme etant s16
s6 est mal reconnu comme etant s1
s6 est mal reconnu comme etant s1
s6 est mal reconnu comme etant s16
s6 est mal reconnu comme etant s1
s6 est mal reconnu comme etant s1
s7 est mal reconnu comme etant s31
s7 est mal reconnu comme etant s37
s7 est mal reconnu comme etant s37
s7 est mal reconnu comme etant s37
s7 est mal reconnu comme etant s37
s8 est mal reconnu comme etant s19
s8 est correctement reconnu
s8 est mal reconnu comme etant s19
s8 est mal reconnu comme etant s19
s8 est correctement reconnu
s9 est mal reconnu comme etant s1
s9 est mal reconnu comme etant s19
s9 est mal reconnu comme etant s19
s9 est mal reconnu comme etant s1
s9 est mal reconnu comme etant s19
Visages correctement reconnus: 36 et mal reconnus: 127

```

Fig. 3: Résultat de l'expérimentation

De ces résultats, nous obtenons un taux de bonne reconnaissance de 18% et un taux de mauvaise reconnaissance de 63.5%. 37 images sont perdues lors de la reconnaissance car ce modèle reconnaissance effectue une détection de visage avant de trouver l'image du modèle correspondante à la nouvelle image en entrée. Nous déduisons des résultats précédemment obtenu que cette approche basée sur opencv n'est pas adapté à nos données. Ceci peut s'expliquer par le fait que pour certaines poses et aux différentes expressions faciales des individus de notre base.

Ci-dessous le résultat d'expérimentation de la solution basée sur le CNN:

```

148/148 [=====] - 5s - loss: 0.0667 - acc: 0.9797 - val_loss:
0.2208 - val_acc: 0.8824
Epoch 47/50
148/148 [=====] - 4s - loss: 0.0232 - acc: 0.9932 - val_loss:
0.2465 - val_acc: 0.8824
Epoch 48/50
148/148 [=====] - 5s - loss: 0.0645 - acc: 0.9865 - val_loss:
0.2042 - val_acc: 0.8824
Epoch 49/50
148/148 [=====] - 4s - loss: 0.0362 - acc: 0.9932 - val_loss:
0.1676 - val_acc: 0.9412
Epoch 50/50
148/148 [=====] - 5s - loss: 0.0215 - acc: 1.0000 - val_loss:
0.1388 - val_acc: 0.9412
Accuracy = 0.941176 and loss = 0.138757

```

Fig. 4: Résultats de l'évaluation obtenus avec CNN.

L'approche implémentée avec le CNN, donne un taux de bonne reconnaissance de 94.1176% au bout de 50 itérations. Nous remarquons une nette amélioration du taux de reconnaissance. Nous pensons que si nous augmentons le nombre de couches et changeons certains paramètres du modèle, nous pourrions augmenter ce taux. Ainsi ce modèle est mieux adapté aux données de *att_faces*.

4- Difficultés rencontrées

La réalisation du projet n'a pas été une partie de plaisir car, plusieurs difficultés ont été rencontrées et nous ont conduit à plusieurs reprises de remettre en doute l'authenticité des systèmes installés dans nos différents pc. Les différents problèmes rencontrés sont entre autres; les bibliothèques de Python qui ne parvenant pas à s'installer; cas de "**theano**" et "**lasagne**". Ces bibliothèques qui n'étaient pas compatibles avec Python 3.6 qui était installé dans nos différentes machines. Ces bibliothèques qui ne parvenaient pas à s'installer nous donnaient une erreur liée à la version du système d'exploitation. Après plusieurs tentatives sans succès, nous avons tous migré à Python 3.5 et à Python 2.7 pas de changement. La solution adéquate est celle de la bibliothèque "**keras**" avec la version **Python 3.5** qui, nous a permis de compiler notre projet sans présenter d'erreur de bibliothèque. Voilà comment nous avons pu contourner cette difficulté.

CONCLUSION

Dans le cadre du module de reconnaissance de forme, il nous a été demandé de mettre en place un système de reconnaissance de visages. Nous avons suivi un ensemble d'étapes avant la réalisation de notre système. Pour la réalisation de cette application, plusieurs exigences ont été présentées à savoir le langage de programmation "Python" et la méthode de classification réseau neuronal convolutionnel (CNN) pour l'extraction des caractéristiques et la classification des visages. Tous ces défis ont été surmontés malgré les difficultés que nous avons rencontrées, nous avons ensuite trouvé des solutions pour une bonne réalisation du projet. Cependant pour effectuer la reconnaissance de visage, notre programme va pouvoir extraire les caractéristiques tels que les yeux, le nez, la bouche et les oreilles. L'approche implémentée avec le CNN, nous a donné un taux de bonne reconnaissance satisfaisant de 94.1176% par rapport à la méthode basique que nous avons implémentée afin de mieux évaluer les performances de notre solution proposée.

LIENS CONSULTÉS

[L1] <https://www.tutorialspoint.com/opencv/index.htm> consulté la dernière fois le 04/06/2017

[L2] https://www.tutorialspoint.com/python3/python_overview.htm consulté la dernière fois le 04/06/2017

[L3] <https://keras.io/getting-started/sequential-model-guide/> dernière consultation 19/06/2017