Université Hassan 1er
École Nationale des Sciences Appliquées – Berrechid

Rapport du projet module Python

Filière Ingénierie des Systèmes D'Information Et Big Data

Réalisation des applications en Deep Learning

- Handwritten Recognition Digits
- Reconnaissance Faciale.

Réalisé par	:	• LAQDOUR Abdelghani • LAGHZAOUI Brahim • ELANSSARI Yassine
Encadré par	:	• Mr. Lahcen MOUMOUN

Année universitaire : 2021/2022

Table des Matières

Listes des figures

Introduction

Aujourd'hui, l'IA est à la fois partout et invisible. Nous la retrouvons dans nos machines informatiques, objets connectés, applications, réseaux sociaux, transports, dans le secteur militaire avec les drones, la publicité et même bientôt dans le réseau mobile. Elle s'essaye aussi dans les domaines plus créatifs comme la musique, le jeu vidéo, l'écriture de fictions ou d'articles. Ensuite L'IA reste limitée à la reconnaissance faciale, l'assistant virtuel, l'automatisation des tâches, la synthèse vocale… C'est ce que l'on retrouve par exemple sur nos smartphones.

La reconnaissance faciale devient peu à peu le moyen le plus sécurisé pour déverrouiller son écran. Bref, ce sont un ensemble de petites choses qui facilitent la vie au quotidien.

Dans ce rapport, nous allons expliquer les étapes de la création de deux application (une application mobile et une application desktop) en quatre chapitres :

- Le chapitre I : Présentation générale.
- Le chapitre II : Handwritten Recognition Digits.
- Le chapitre III : la reconnaissance faciale.
- Le chapitre IV : Environnement et outils de travail.
- Le chapitre V : La réalisation du projet

Chapitre I :Présentation générale

1 L’intelligence Artificiel

L'intelligence démontrée par les machines est connue sous le nom d'intelligence artificielle. L'intelligence artificielle est devenue très populaire dans le monde d'aujourd'hui. C'est la simulation de l'intelligence naturelle dans des machines qui sont programmées pour apprendre et imiter les actions des humains. Ces machines sont capables d'apprendre avec l'expérience et d'effectuer des tâches humaines. À mesure que des technologies telles que l'IA continue de se développer, elles auront un impact considérable sur notre qualité de vie. Il est naturel que tout le monde veuille aujourd'hui se connecter d'une manière ou d'une autre à la technologie de l'IA, que ce soit en tant qu'utilisateur final ou poursuivant une carrière dans l'intelligence artificielle.

2 . Machine Learning :

Le Machine Learning est une technologie d’intelligence artificielle permettant aux ordinateurs d’apprendre sans avoir été programmés explicitement à cet effet. Pour apprendre et se développer, les ordinateurs ont toutefois besoin de données à analyser et sur lesquelles s’entraîner. De fait, le Big Data est l’essence du Machine Learning, et c’est la technologie qui permet d’exploiter pleinement le potentiel du Big Data. Découvrez pourquoi cette technique et le Big Data sont interdépendants.

Si le Machine Learning ne date pas d’hier, sa définition précise demeure encore confuse pour de nombreuses personnes. Concrètement, il s’agit d’une science moderne permettant de découvrir des patterns et d’effectuer des prédictions à partir de données en se basant sur des statistiques, sur du forage de données, sur la reconnaissance de patterns et sur les analyses prédictives. Les scientifiques des données peuvent utiliser toutes ces données pour former des modèles Machine Learning capables de faire des prédictions et des inférences en fonction des relations qu’ils découvrent dans les données.

Le Machine Learning peut être défini comme une branche de l’intelligence artificielle englobant de nombreuses méthodes permettant de créer automatiquement des modèles à partir des données. Ces méthodes sont en fait des algorithmes.

a. L’apprentissage supervisé :

Ici, la machine s’appuie sur des classes prédéterminées et sur un certain nombre de paradigmes connus pour mettre en place un système de classement à partir de modèles déjà catalogués. Dans ce cas, deux étapes sont nécessaires pour compléter le processus, à commencer par le stade d’apprentissage qui consiste à la modélisation des données cataloguées. Ensuite, il s’agira au second stade de se baser sur les données ainsi définies pour attribuer des classes aux nouveaux modèles introduits dans le système, afin de les cataloguer eux aussi. Le Machine Learning peut être défini comme **une branche de l’intelligence artificielle** englobant de nombreuses méthodes permettant de créer automatiquement des modèles à partir des données. Ces méthodes sont en fait des algorithmes.
Parmi les algorithmes supervisés, on distingue **les algorithmes de classification** (prédictions non-numériques) et **les algorithmes de régression** (prédictions numérique). En fonction du problème à résoudre, on utilisera l’un de ces deux archétypes.

b. L’apprentissage non supervisé :

L’**apprentissage non supervisé**, au contraire, consiste à entraîner le modèle sur des données sans étiquettes. La machine parcourt les données sans aucun indice, et tente d’y découvrir des motifs ou des tendances récurrents. Cette approche est couramment utilisée dans certains domaines, comme la cybersécurité.
Parmi les modèles non-supervisés, on distingue **les algorithmes de clustering** (pour trouver des groupes d’objets similaires), **d’association** (pour trouver des liens entre des objets) et **de réduction dimensionnelle** (pour choisir ou extraire des caractéristiques).

3 . Deep Learning :

Le Deep Learning, ou apprentissage profond, est l’une des principales technologies de Machine Learning et d’intelligence artificielle.

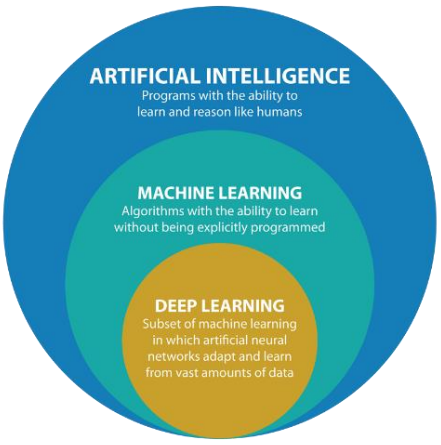


Figure 1 : Deep learning, ML , IA

Différentes techniques de Machine Learning ont été développées pour créer des algorithmes capables d'apprendre et de s'améliorer de manière autonome.
Parmi ces techniques, **on compte les réseaux de neurones artificiels**. C'est sur ces algorithmes que reposent le Deep Learning, mais aussi des technologies comme la reconnaissance d'images ou la vision robotique.

a- Les réseaux de neurones :

Les réseaux de neurones artificiels sont inspirés par les neurones du cerveau humain. Ils sont constitués de plusieurs neurones artificiels connectés entre eux. Plus le nombre de neurones est élevé, plus le réseau est « profond ».

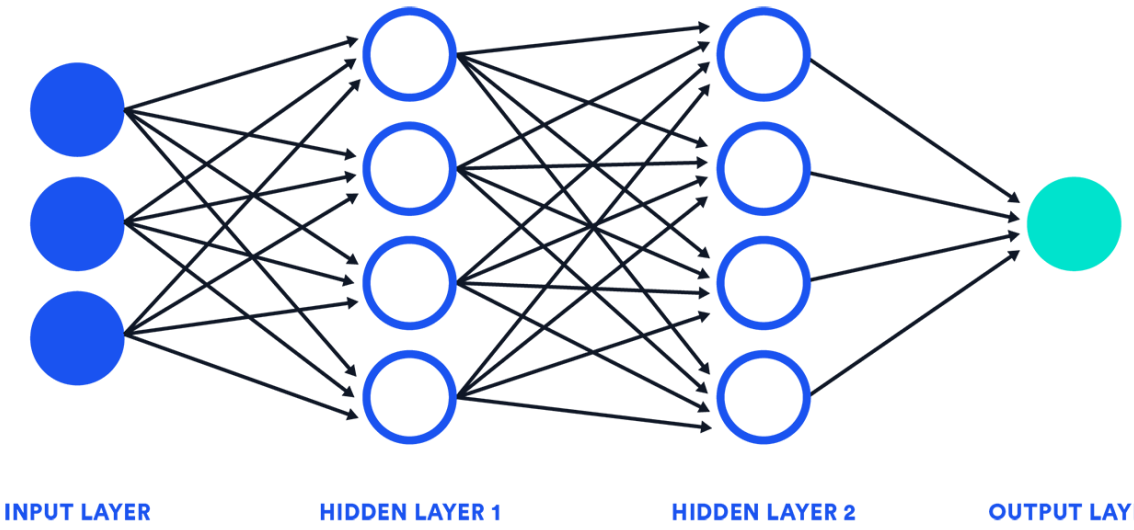


Figure 2 : Réseau de neurones

b- Réseau de neurones convolutifs

Un convolution neural network (CNN) est un type de réseau neuronal artificiel utilisé dans la reconnaissance et le traitement d'images et spécifiquement conçu pour traiter les données de pixels.

Un CNN utilise un système semblable à un perceptron multicouche qui a été conçu pour des besoins de traitement réduits. Les couches d'un CNN se composent d'une couche d'entrée, d'une couche de sortie et d'une couche cachée qui comprend plusieurs couches convolutionnelles, des couches de regroupement, des couches entièrement connectées et des couches de normalisation. La suppression des limitations et l'augmentation de l'efficacité pour le traitement des images aboutissent à un système beaucoup plus efficace, plus simple à former, et spécialisé pour le traitement des images et le traitement du langage naturel.

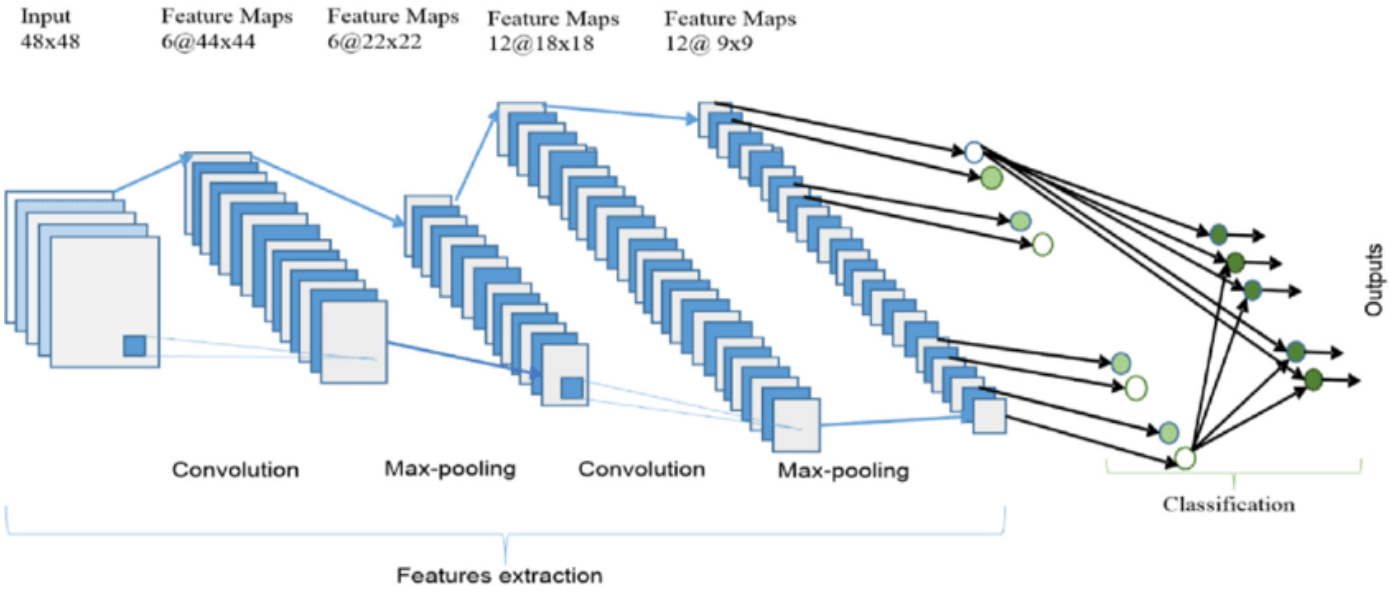


Figure 3 : Architecture de CNN

Chapitre II :Handwritten Recognition Digits

1- Objectifs :

Le but de ce projet est de concevoir un réseau de neurone de convolution afin de classifier une image. Les premiers tests se feront sur la base de données MNIST qui comporte des images manuscrites de chiffres. Une fois que le réseau a appris, on lui soumet une image d'un chiffre qu'il n'a pas vu lors de son apprentissage, et il doit produire en sortie le chiffre reconnu.

2- DataSet MNIST :

C'est une Dataset de chiffres écrits à la main. C'est un jeu de données très utilisé en apprentissage automatique.

La reconnaissance de l'écriture manuscrite est un problème difficile, et un bon test pour les algorithmes d'apprentissage. La base MNIST est devenue un test standard. Elle regroupe 60,000 images d'apprentissage et 10,000 images de test, issues d'une base de données antérieure, appelée simplement NIST. Ce sont des images en noir et blanc, normalisées centrées de 28 pixels de côté.

MNIST est une base de données étiquetée propice pour un apprentissage supervisé. Dans l'image ci-dessus, pour chaque chiffre, on a sa représentation sous forme d'image ainsi que son étiquette. Par exemple, pour le dernier chiffre en bas à droit, l'étiquette vaut 9 vu qu'il s'agit du chiffre 9. La représentation de ces chiffres est normalisée à travers tout le jeu de données MNIST. Ainsi, chaque chiffre est codé dans un format 8 pixels * 8 pixels. En plus, chaque pixel peut prendre une valeur de 0 à 255. Cette plage de valeurs représente le niveau de gris Grayscale. En d'autres termes, chaque représentation **d'une image est une matrice** de dimension 8 x 8.

Le jeu de données MNIST présent par défaut dans la librairie Scikit Learn, comporte un sous-ensemble de la "vraie" base de données MNIST. Le sous-ensemble comporte 1797 chiffres que nous diviserons par la suite en deux sous-ensembles : d'entrainement et de test.

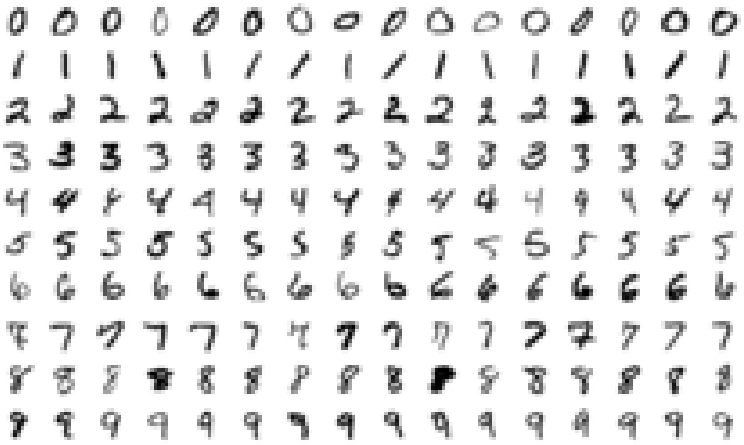


Figure 4 : DataSet MNIST

3- Modèle Utilisé :

Nous expliquons ici les différentes étapes pour préparer notre modèle.

✓ Importation des librairies

Premièrement, on doit importer toutes les bibliothèques nécessaires pour construire le model et l'entrainer.

✓ Chargement de Dataset MNIST

On va utiliser la Dataset qui se trouve dans la bibliothèque KERAS, pour le charger on utilise la fonction suivante :

```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

Figure 5 : chargement de MNIST Dataset

Ici on a divisé les données en deux parties :

- Données d'entraînement (90 %) : qui contient Y_train qui représentant les labels et X_train les images sous format des matrices des pixel et qui sont utilisé pour entraîner le modèle.
- Des données de test (10 %) : Y_test et X_test pour d'effectuer une évaluation de notre modèle.

✓ Redimensionner et normaliser les données

- Redimensionner l'image en 3 dimensions (hauteur = 28px, largeur = 28px, canal = 1).
  - ⇨ Canal = 1 => Pour l'échelle de gris avec reshape ().
- Nous effectuons une normalisation en niveaux de gris réduire la complexité de modèle. De plus les CNN convergent plus rapidement sur les données [0..1] que sur [0..255].

✓ Définition du modèle

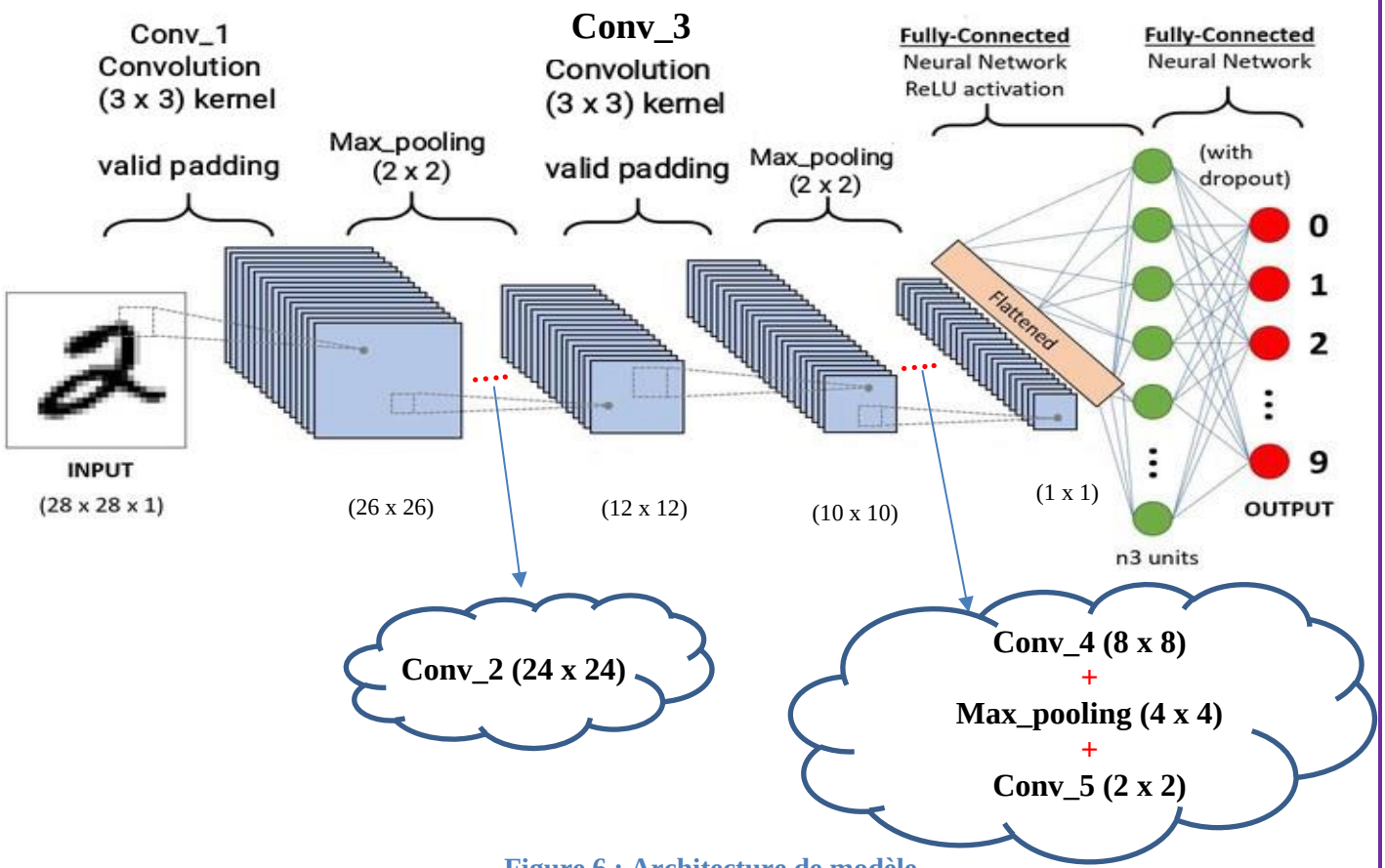


Figure 6 : Architecture de modèle

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 64)        640

 conv2d_1 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 12, 12, 64)       0
 )

 conv2d_2 (Conv2D)           (None, 10, 10, 128)       73856

 conv2d_3 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 conv2d_4 (Conv2D)           (None, 2, 2, 256)         295168

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 flatten (Flatten)           (None, 256)               0

 batch_normalization (BatchN  (None, 256)              1024
 ormalization)

 dense (Dense)               (None, 512)               131584

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 691,914
Trainable params: 691,402
Non-trainable params: 512
```

Figure 7 : description du modèle

La couche convolutive_1 (Conv2D) :

Dans cette couche on a choisi 64 filtres de taille (3 x 3) c'est-à-dire chaque image de dimension (28 x 28) on va le multiplier par chaque filtres (multiplier image par 64 filtres) à la fin on va trouver 64 images avec une dimension de (26 x 26).

La couche convolutive_2 (Conv2D) :

Dans cette couche on a choisi aussi 64 filtres de taille (3 x 3), on va multiplier chaque image de dimension (26 x 26) par chaque filtres (multiplier image par 64 filtres) à la fin on va trouver 4096 images avec une dimension de (24 x 24).

La couche Max_pooling_1 :

Cette couche agit simplement comme un filtre de sous-échantillonnage. Il regarde les 2 pixels voisins et choisit la valeur maximale. Ceux-ci sont utilisés pour réduire les coûts de calcul et, dans une certaine mesure, réduire également le surapprentissage. Il faut choisir la taille de pooling (c'est-à-dire

la taille de la zone mutualisée à chaque fois) plus la dimension de pooling est élevée, plus le downsampling est important.

Ici on va prendre chaque images (format matrice) et on va applique le max pooling c'est-à-dire on va prendre la valeur max dans chaque sous matrice (la dimension de sous matrice on peut le définir, dans notre cas c'est (2 x 2)) comme vous voyez dans l'image ci-dessous.

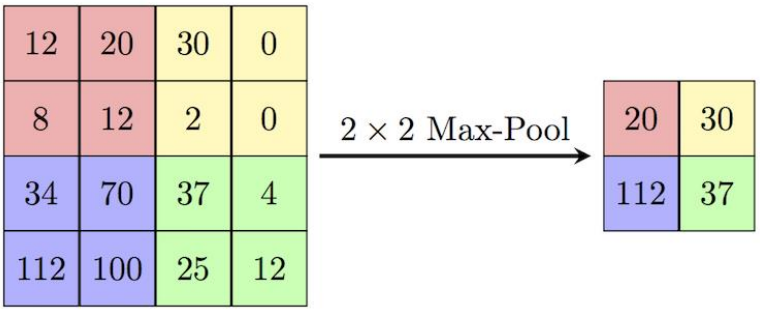


Figure 8 : la couche de pooling

La dimension des images après l'application de Max_pooling est (12 x 12 x 1) c'est-à-dire 24/2, et nombre des images c'est le même 4096.

La couche convolutive_3 (Conv2D) :

Dans cette couche on a choisi 128 filtres de taille (3 x 3) c'est-à-dire chaque image de dimension (12 x 12) on va le multiplier par chaque filtres (multiplier image par 128 filtres) et à la fin nous constaterons que le nombre d'images est 524288 avec une dimension de (10 x 10).

La couche convolutive_4 (Conv2D) :

Dans cette couche on a choisi aussi 128 filtres de taille (3 x 3), on va multiplier chaque image de dimension (10 x 10) par chaque filtres (multiplier image par 128 filtres) à la fin on va trouver 67108864 images avec une dimension de (8 x 8).

La couche Max_pooling_2 :

Nous appliquerons le max pooling pour chaque images (format matrice) c'est-à-dire que nous prendrons la valeur maximale dans chaque sous matrice et à la fin nous constaterons que le nombre d'images restera le même et que les dimensions des images sont : (4 x 4)

La couche convolutive_5 (Conv2D) :

Dans cette couche on a choisi 256 filtres de taille (3 x 3) c'est-à-dire chaque image de dimension (4 x 4) on va le multiplier par chaque filtres (multiplier image par 256 filtres) à la fin on va trouver 17 179 869 184 images avec une dimension de (2 x 2).

La couche Max_pooling_2 :

Ici, nous ferons la même chose que l'autre couch de Max_pooling et à la fin on va trouver que le nombre d'images restera le même et que les dimensions des images sont : (1 x 1)

⇨ Dans les couch précédant on a utilisé la fonction d'activation 'RELU' :

Est le redresseur (fonction d'activation max (0, x). La fonction d'activation du redresseur est utilisée pour ajouter de la non linéarité au réseau. (Supprimer les nombres négatifs)

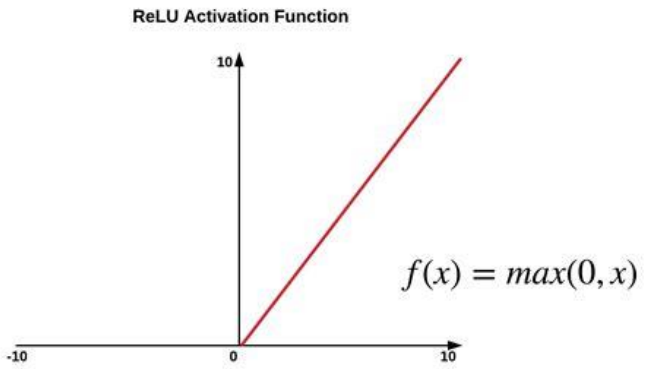


Figure 9 : La fonction RELU

La couch Flatten :

La couche Flatten est utilisée pour convertir les cartes de caractéristiques finales en un seul vecteur 1D. Cette étape d'aplatissement est nécessaire pour que vous puissiez utiliser des couches entièrement connectées après certaines couches convolutives/maxpool. Il combine toutes les caractéristiques locales trouvées des couches convolutives précédentes.

Les autre couch de réseaux de neurones (ANN) :

En fin de compte, Nous allons utiliser les fonctionnalités dans deux couches entièrement connectées

(Dense) qui n'est qu'un classificateur artificiel de réseaux de neurones (ANN).

Dans la dernière couche (Dense (10, activation="softmax")), le réseau génère la distribution de probabilité de chaque classe.

Et la couche de sortie a 10 neurones pour les 10 classes.

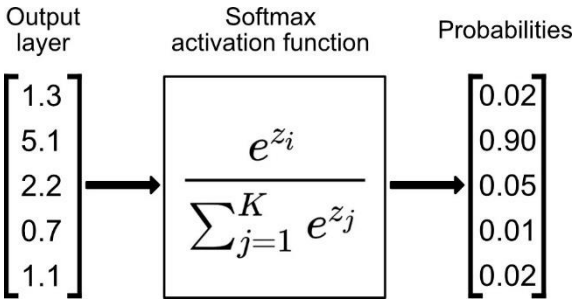


Figure 10 : La fonction d'activation SoftMax

✓ Compilation de model :

Nous allons compiler notre modèle avec la fonction d'optimiseur ADAM (Une extension de la descente de gradient stochastique qui a récemment été largement adoptée pour les applications d'apprentissage en profondeur dans les domaines de la vision par ordinateur).

✓ Entrainement de modèle

C'est maintenant l'étape la plus **coûteuse en temps machine** : le réseau de neurone va s'entraîner pour la tâche qu'on essaie de lui apprendre.

C'est le rôle de La fonction **model.fit ()** de KERAS il faut donc lui fournir, entre autres :

- Des données d'entrée avec les labels correspondants.
- Des données de validation, pour vérifier un certain nombre de chose (overfitting, ...).
- Le nombre d'epochs que va durer l'apprentissage.

```
Epoch 1/10
525/525 [==============================] - 179s 338ms/step - loss: 0.1030 - accuracy: 0.9685 - val_loss: 0.1065 - val_accuracy:
0.9655
Epoch 2/10
525/525 [==============================] - 178s 340ms/step - loss: 0.0424 - accuracy: 0.9865 - val_loss: 0.0545 - val_accuracy:
0.9839
Epoch 3/10
525/525 [==============================] - 171s 326ms/step - loss: 0.0363 - accuracy: 0.9890 - val_loss: 0.0366 - val_accuracy:
0.9911
Epoch 4/10
525/525 [==============================] - 179s 341ms/step - loss: 0.0254 - accuracy: 0.9922 - val_loss: 0.0380 - val_accuracy:
0.9911
Epoch 5/10
525/525 [==============================] - 187s 356ms/step - loss: 0.0258 - accuracy: 0.9918 - val_loss: 0.0587 - val_accuracy:
0.9865
Epoch 6/10
525/525 [==============================] - 167s 317ms/step - loss: 0.0195 - accuracy: 0.9935 - val_loss: 0.0485 - val_accuracy:
0.9890
Epoch 7/10
525/525 [==============================] - 168s 320ms/step - loss: 0.0184 - accuracy: 0.9943 - val_loss: 0.0978 - val_accuracy:
0.9804
Epoch 8/10
525/525 [==============================] - 172s 328ms/step - loss: 0.0203 - accuracy: 0.9937 - val_loss: 0.0463 - val_accuracy:
0.9900
Epoch 9/10
525/525 [==============================] - 165s 315ms/step - loss: 0.0149 - accuracy: 0.9953 - val_loss: 0.0550 - val_accuracy:
0.9861
Epoch 10/10
525/525 [==============================] - 162s 309ms/step - loss: 0.0125 - accuracy: 0.9962 - val_loss: 0.0456 - val_accuracy:
0.9906
```

Figure 11 : Entrainement de modèle

Quand le modèle termine le processus d'entrainement, nous allons enregistrer le modèle dans un fichier nommé model.h5 qu'on va l'utiliser dans notre application mobile.

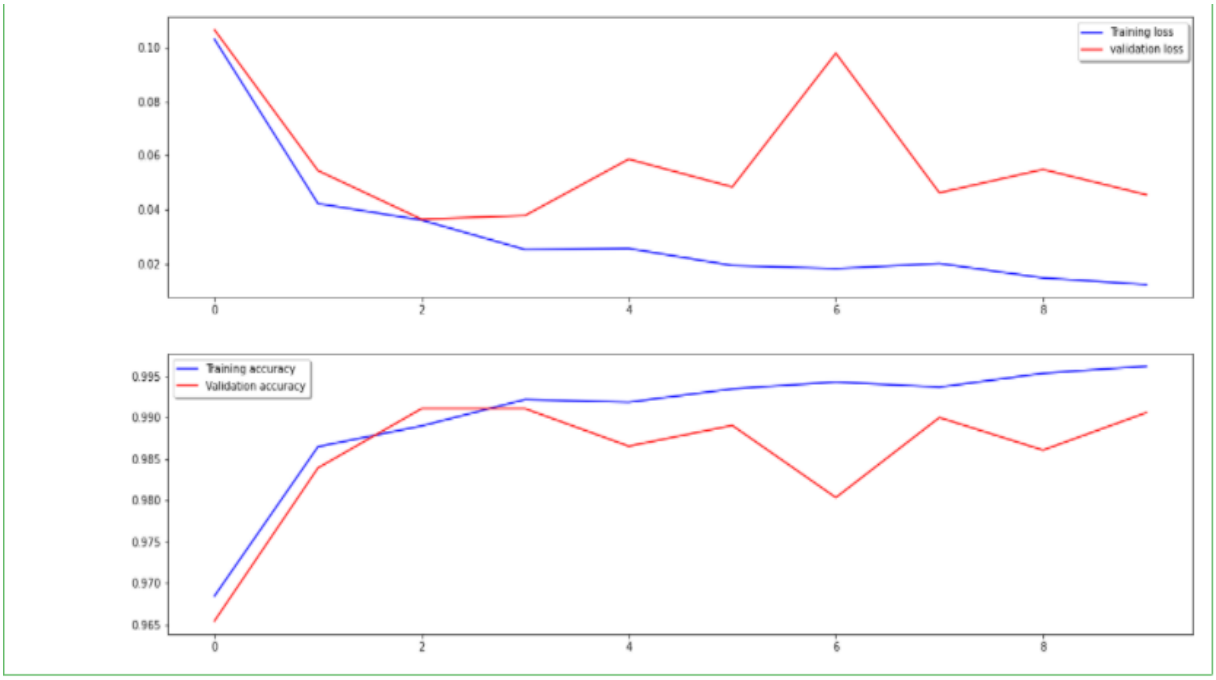


Figure 12 : Les courbes de perte et de précision pour la formation et la validation

✓ Résultats de la validation des prédictions :

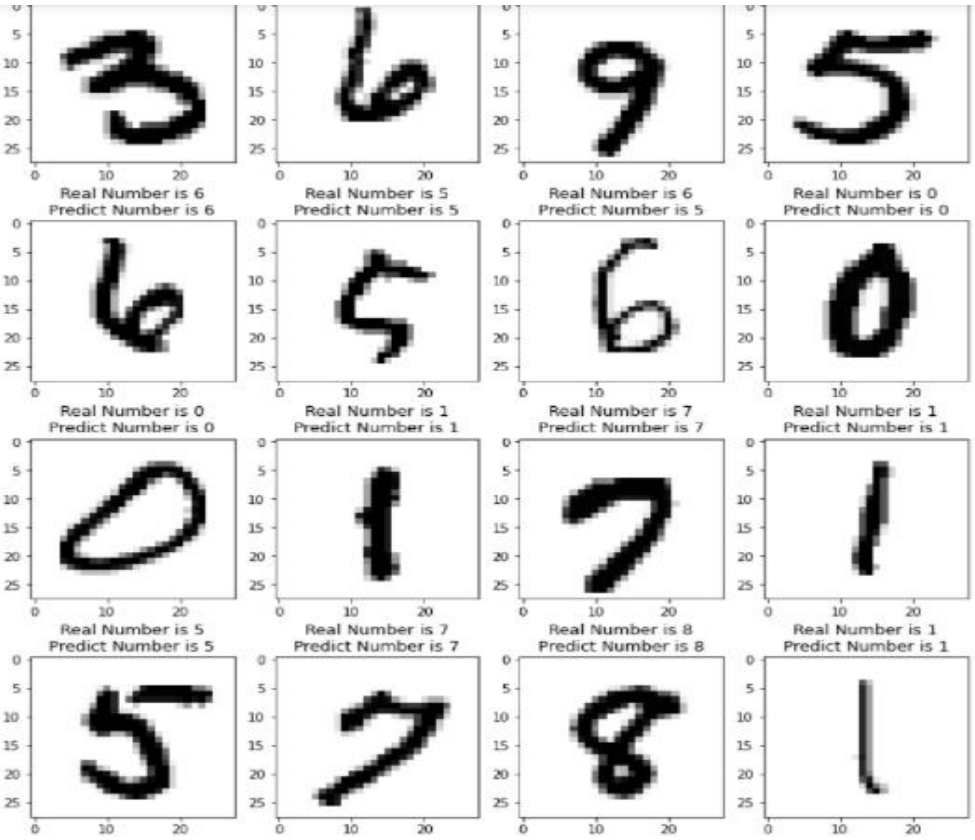


Figure 13 : Résultats de la prédiction

Chapitre III : La reconnaissance faciale

1- Introduction

L'utilisation des techniques de reconnaissance faciale a connu un développement à grande échelle depuis le milieu des années 90, avec l'utilisation efficace de nouvelles technologies, notamment l'ordinateur et sa capacité de traitement d'images. L'utilisation de ces techniques existe depuis qu'une machine est capable de comprendre ce qu'elle « voit » lorsqu'on la connecte à une ou plusieurs caméras, c'est à dire que les premiers essais datent du début des années 70. et sont basés sur des méthodes à bases d'heuristiques, basés sur des attributs faciaux mesurables comme l'écartement des yeux, des sourcils, des lèvres, la position du menton, la forme, etc.

1- Objectifs :

Notre Objective consiste à créer une application qui permettant de capter une image d'une personne et en comparant son visage avec ceux qui on a déjà dans notre Dataset afin d'afficher les 5 premières personnes similaires a lui.

4- Dataset :

Dataset que nous avons utilisé contenant des images de 28 personnes entre des acteurs (brad pitt, angelina jolie…) et des sportifs (Messi, Ronaldo, Mohamed Ali...), qui sont structurer dans un dossier '' Dataset'' et pour chaque personne un dossier qui contenant ses images.

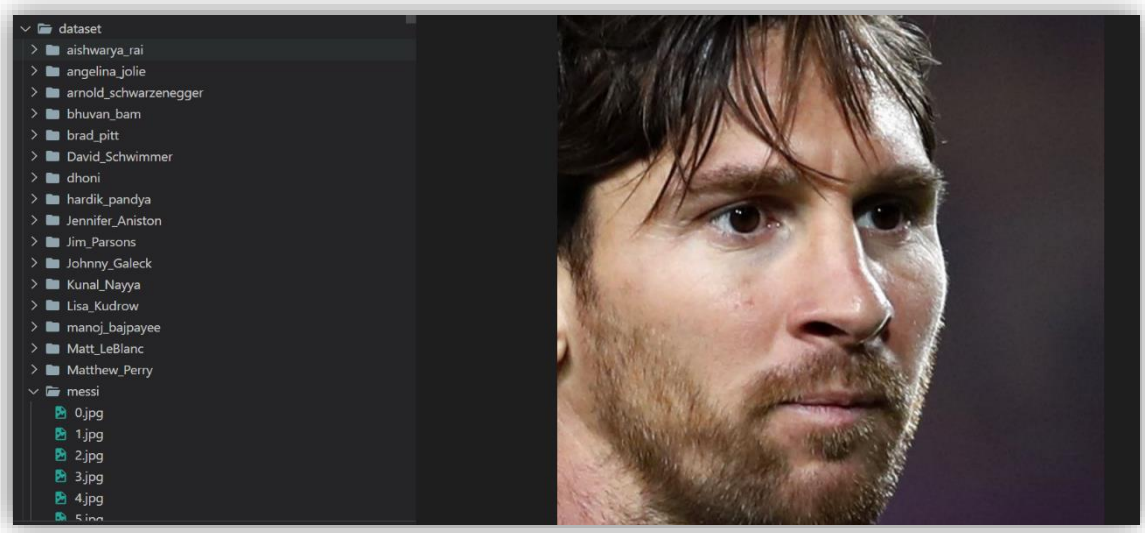


Figure 14 : DataSet de célébrités

Preparation du dataset :

Dataset, elle comprend un total de 9364 images RGB, recadrées de (96,96,3) pixels de visages chacun étiqueté avec l'une des 28 classes personnes suivantes :

0 → 418 images : aishwarya_rai.

1 → 394 images : angelina_jolie.

2 → 349 images : arnold_schwarzenegger.

•

•

•

25 → 329 images : suresh_raina.

26 → 287 images : sylvester_stallone.

27 → 391 images : virat_kohli.

Qui sont reparties en deux parties à savoir :

- ⇨ 1405 images représentant les images de test (15%).
- ⇨ 7959 représentent les images d'apprentissage (75%).

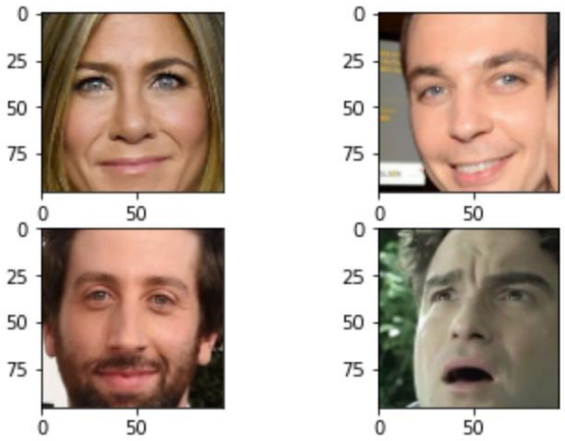


Figure 15 : Echantillon d'image

⇨ Pour les redimensionnant entre -1 et 1, les images sont redimensionnées à [0,1] en le divisant par 255.

5- Modèle Utilisé :

✓ Architecture de Modèle

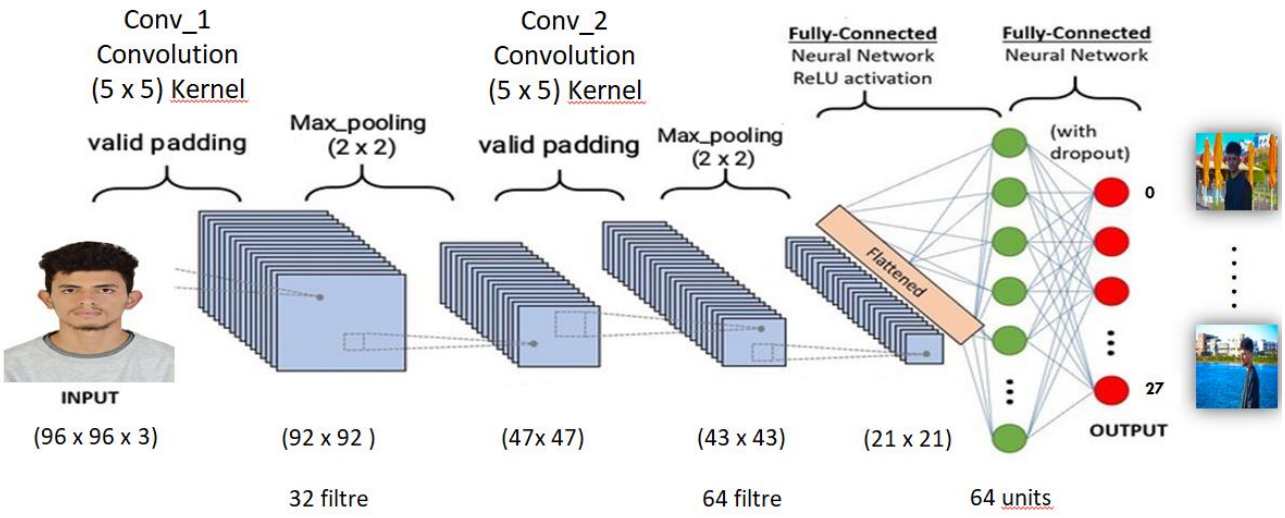
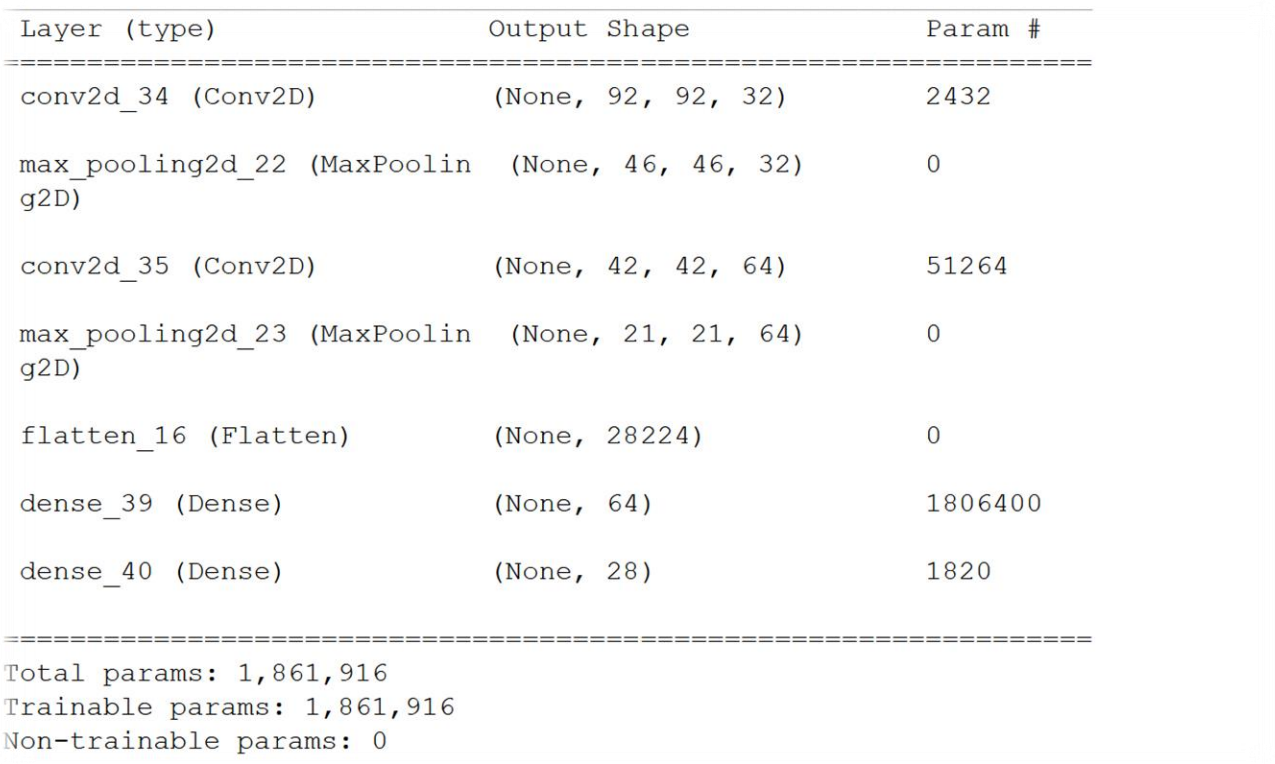


Figure 16 : Architecture de modèle

Après la préparation des données, on a procédé à l'apprentissage pour l'extraction des caractéristiques des entrées en utilisant l'architecture CNN. Il est constitué de deux couches de convolutions chacune d'elle est suivie par une fonction d'activation **tanh** et fonction **Maxpooling** de taille (2,2). Après l'extraction des caractéristiques des entrées, on attache à la fin du réseau un perceptron ou bien un MLP (multi layer perceptron) qui est constitué une couche d'entrée de 64 neurones et à la fin produit un vecteur de 28 dimensions ou 28 est le nombre de classe (nombre des personnes) ou chaque élément est la probabilité d'appartenance à une classe (de chaque personne qui similaire a image entrée). Chaque probabilité est calculée à l'aide de la fonction **softmax**.



Layer (type)	Output Shape	Param #
conv2d_34 (Conv2D)	(None, 92, 92, 32)	2432
max_pooling2d_22 (MaxPooling2D)	(None, 46, 46, 32)	0
conv2d_35 (Conv2D)	(None, 42, 42, 64)	51264
max_pooling2d_23 (MaxPooling2D)	(None, 21, 21, 64)	0
flatten_16 (Flatten)	(None, 28224)	0
dense_39 (Dense)	(None, 64)	1806400
dense_40 (Dense)	(None, 28)	1820

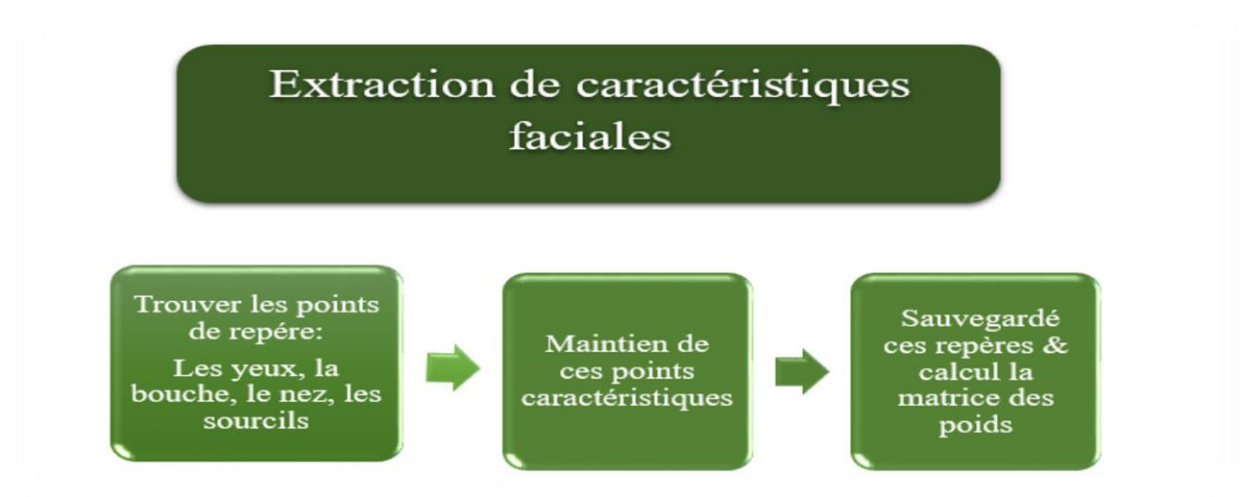
Total params: 1,861,916
Trainable params: 1,861,916
Non-trainable params: 0

Figure 17 : Description de modèle

✓ Extraction de caractéristiques faciales :

Une fois le visage capté, 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 utilisée 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 cœur du système de reconnaissance, on extrait de l'image les Données qui seront sauvegardées en mémoire pour être utilisées plus tard dans la phase de décision (les 5 premières personnes similaires à l'image).



✓ Compilation de model :

Nous allons compiler notre modèle avec la fonction d'optimiseur ADAM (Une extension de la descente de gradient stochastique qui a récemment été largement adoptée pour les applications d'apprentissage en profondeur dans les domaines de la vision par ordinateur).

✓ Entrainement de modèle

La fonction principale qui est responsable de l'apprentissage c'est la fonction fit () avec les paramètres suivants : Données d'entraînement (X_train), données cibles (Y_train), données de validation et nombre d'époques. Pour nos données de validation, nous utilisons l'ensemble de test fourni dans notre ensemble de données, que nous avons divisé en X_test et Y_test. X_train et X_train constituent les données d'image elles-mêmes, tandis que Y_train et Y_test constituent les étiquettes.

```
Epoch 1/10
19/19 [==============================] - 108s 5s/step - loss: 1.9294 - accuracy: 0.4742 - val_loss: 1.8660 - val_accuracy:
0.4992
Epoch 2/10
19/19 [==============================] - 102s 5s/step - loss: 1.6926 - accuracy: 0.5514 - val_loss: 1.6941 - val_accuracy:
0.5502
Epoch 3/10
19/19 [==============================] - 104s 6s/step - loss: 1.5393 - accuracy: 0.5974 - val_loss: 1.5857 - val_accuracy:
0.5672
Epoch 4/10
19/19 [==============================] - 100s 5s/step - loss: 1.4254 - accuracy: 0.6274 - val_loss: 1.5005 - val_accuracy:
0.5868
Epoch 5/10
19/19 [==============================] - 98s 5s/step - loss: 1.3429 - accuracy: 0.6499 - val_loss: 1.4530 - val_accuracy: 0.
6038
Epoch 6/10
19/19 [==============================] - 99s 5s/step - loss: 1.2682 - accuracy: 0.6732 - val_loss: 1.3955 - val_accuracy: 0.
6249
Epoch 7/10
19/19 [==============================] - 100s 5s/step - loss: 1.1990 - accuracy: 0.6891 - val_loss: 1.3366 - val_accuracy:
0.6455
Epoch 8/10
19/19 [==============================] - 97s 5s/step - loss: 1.1409 - accuracy: 0.7074 - val_loss: 1.3052 - val_accuracy: 0.
6522
Epoch 9/10
19/19 [==============================] - 101s 5s/step - loss: 1.0943 - accuracy: 0.7175 - val_loss: 1.2586 - val_accuracy:
0.6631
Epoch 10/10
19/19 [==============================] - 96s 5s/step - loss: 1.0407 - accuracy: 0.7334 - val_loss: 1.2312 - val_accuracy: 0.
6775
```

Figure 18 : Entrainement de modèle

On peut voir que le réseau s'est entraîné pendant 10 époques et nous avons atteint une grande précision (73,34%) et une faible perte qui suit la perte d'entraînement, comme le montre l'image.

- **Les courbes de perte et de précision pour la formation et la validation :**

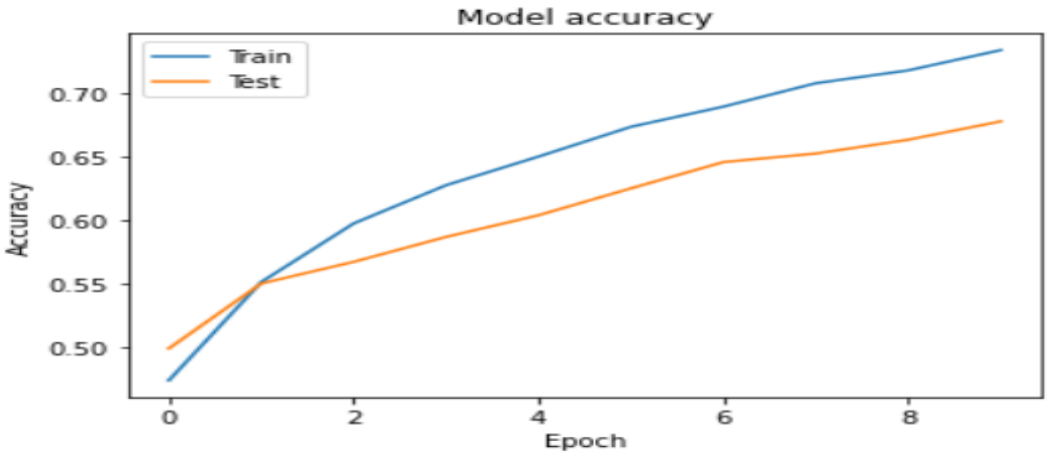


Figure 19 : Courbe de précision

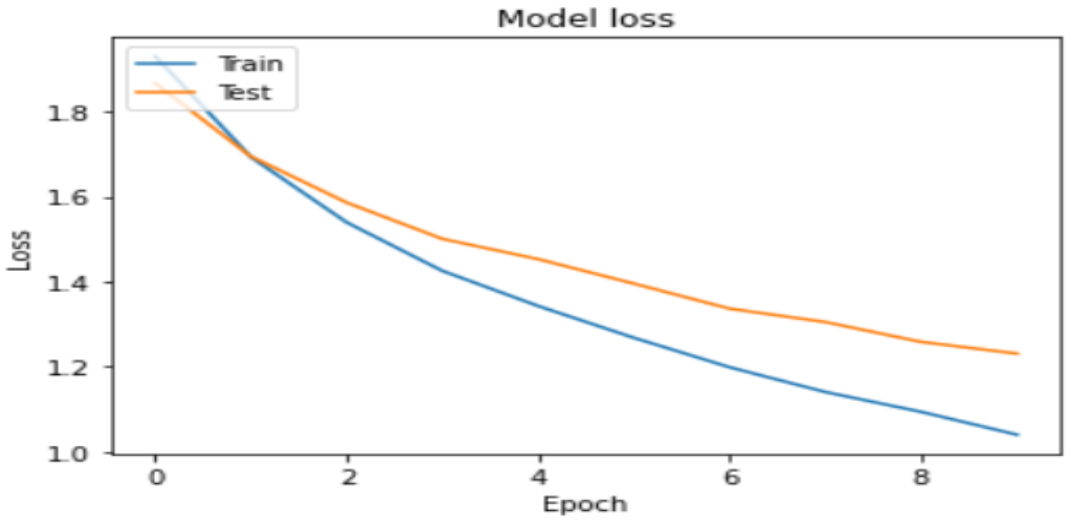


Figure 20 : Courbe de Perte

Après l'analyse des résultats obtenus, On constate les remarques suivantes : La précision de l'apprentissage et de test augmente avec le nombre d'époque, ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. Si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre d'époque et vice versa. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

Chapitre IV : Environnement et outils de travail

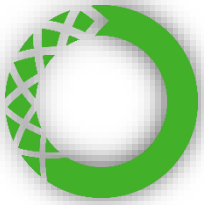
✓ Flutter

Flutter c'est un Framework basé sur langage Dart (on va le voir ci-dessous) gratuit et source de développement natif multiplateforme créé par Google. Flutter prend en change le mobile (Android & IOS), le desktop (Windows, Mac OS, Linux, etc.), les périphériques intégrés (Raspberry Pi, Google Home Hub etc.) Et le web.

✓ Android Studio

Android Studio est l'environnement de développement intégré (IDE) officiel du système d'exploitation Android de Google, construit sur le logiciel JetBrains IntelliJ IDEA et conçu spécifiquement pour le développement Android.

✓ Anaconda



Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique, qui vise à simplifier la gestion des paquets et de déploiement

✓ Jupyter

Le notebook Jupyter est un environnement HTML pour Python, Il fournit un environnement organisé en cellules interactives qui peuvent être exécutées, permettant l'organisation et la documentation de calculs de façon structurée.

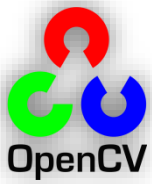
✓ TensorFlow

TensorFlow est une outil open source de bout en bout pour l'apprentissage automatique. Il dispose d'un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires qui permet aux chercheurs de pousser l'état de l'art en matière de ML.

✓ KERAS

La bibliothèque Keras permet d'interagir avec les algorithmes de réseaux de neurones profonds et d'apprentissage automatique.

✓ OpenCV



Est une bibliographique et libre, initialement développé par Intel, spécialisée dans le traitement d'images en temps réel.

✓ Tkinter

Tkinter est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques

✓ Python

Python est un langage de programmation interprété et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions.

Chapitre V : La réalisation du projet

1- L’application de Handwritten Recognition Digits

○ Page d’accueil :

Sur cette page, nous n'avons qu'un message de bienvenue ainsi qu'un Button pour accéder à l'application.

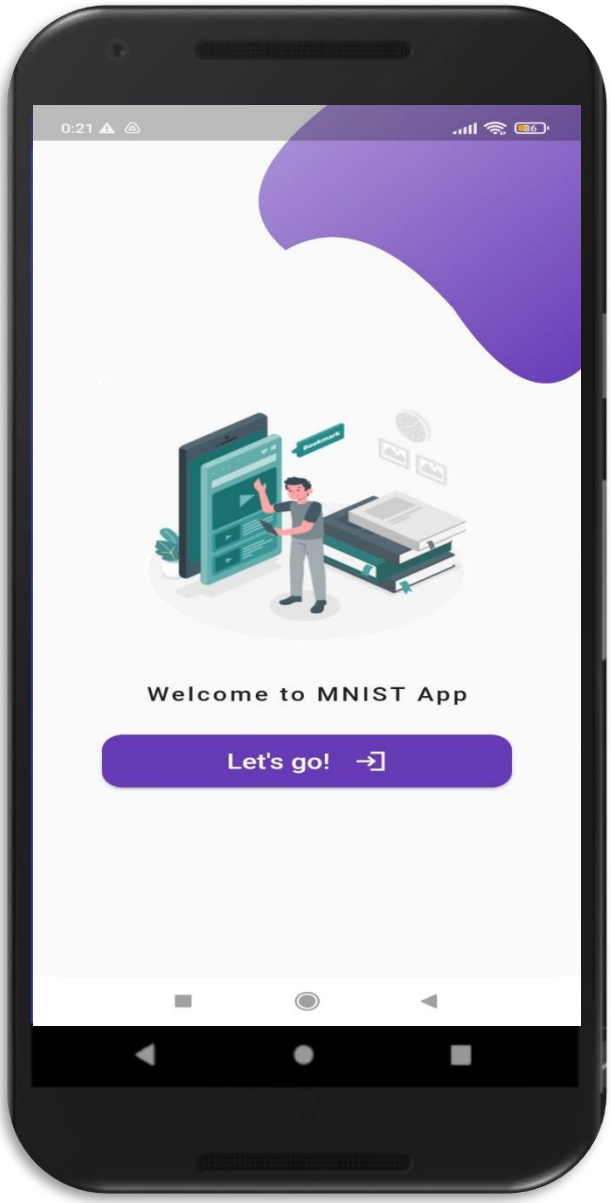


Figure 21 : Page d’accueil

○ Page MNIST :

Dans cette page nous avons deux interfaces :

⇨ **La première interface :** pour importer une image et voir leur prédiction.

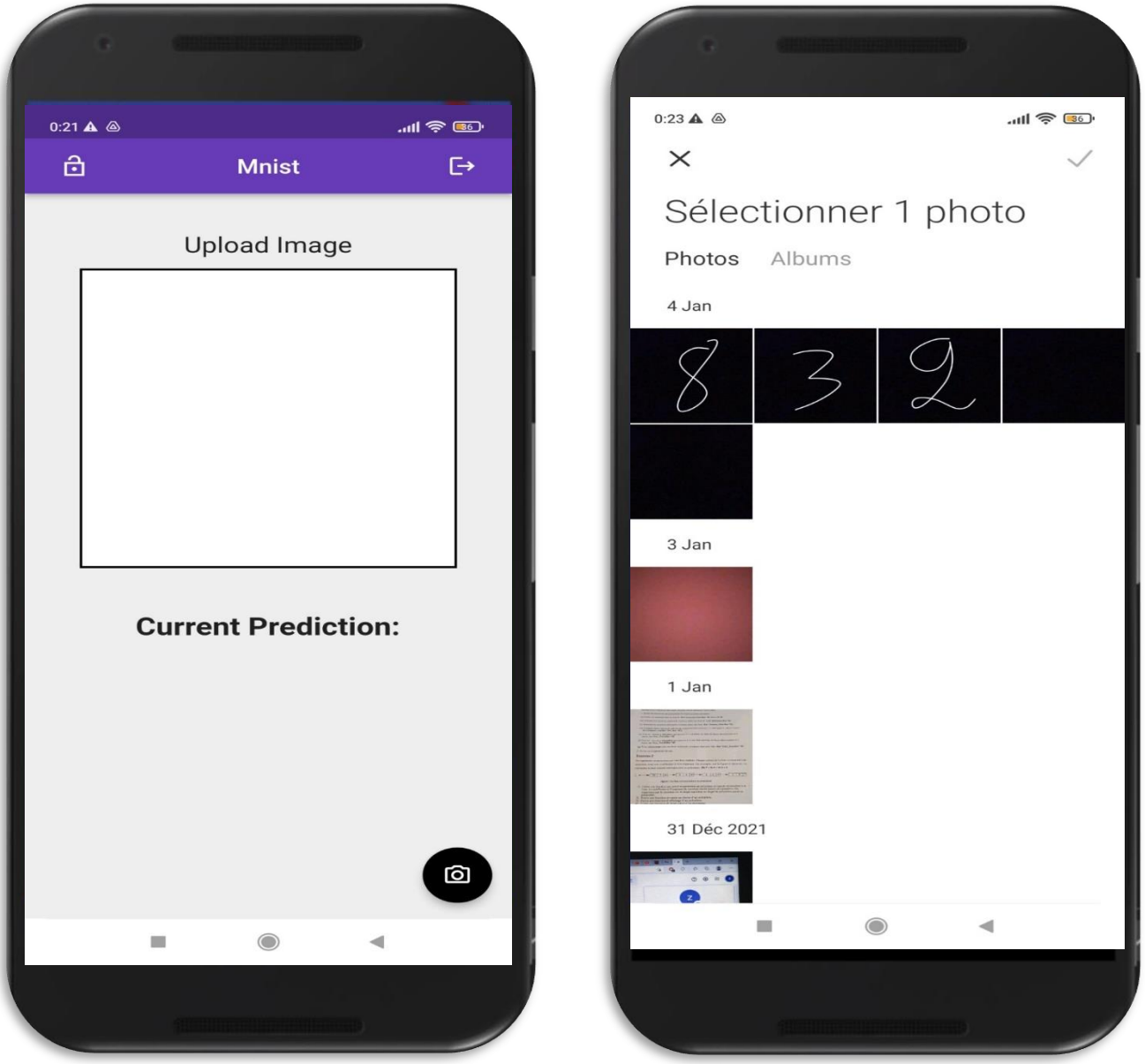


Figure 22 : Page Mnist

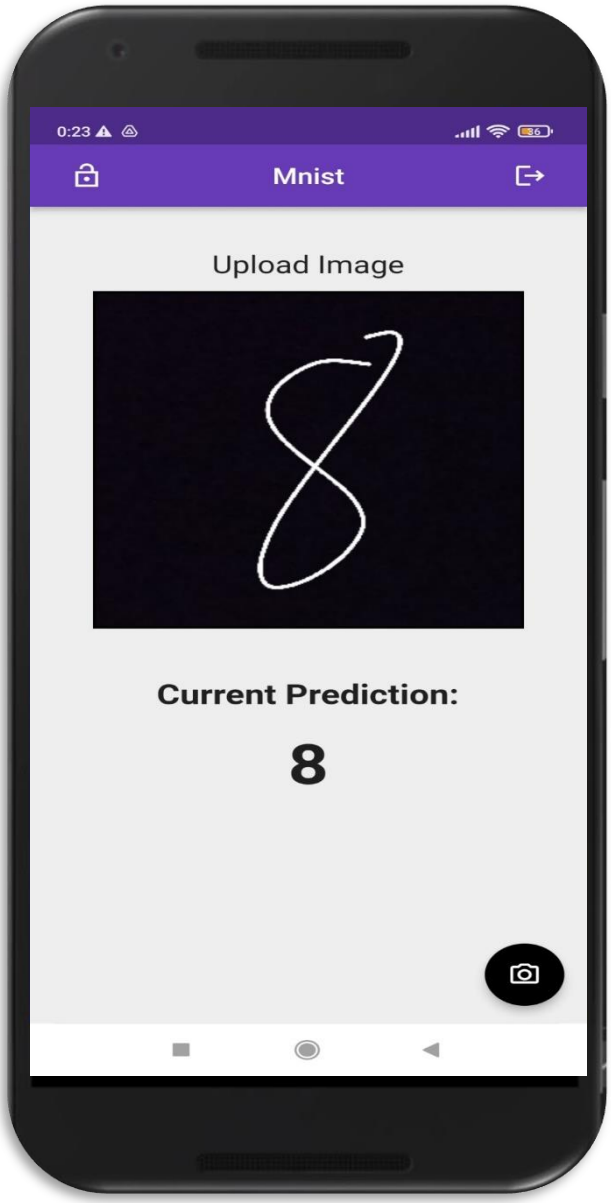
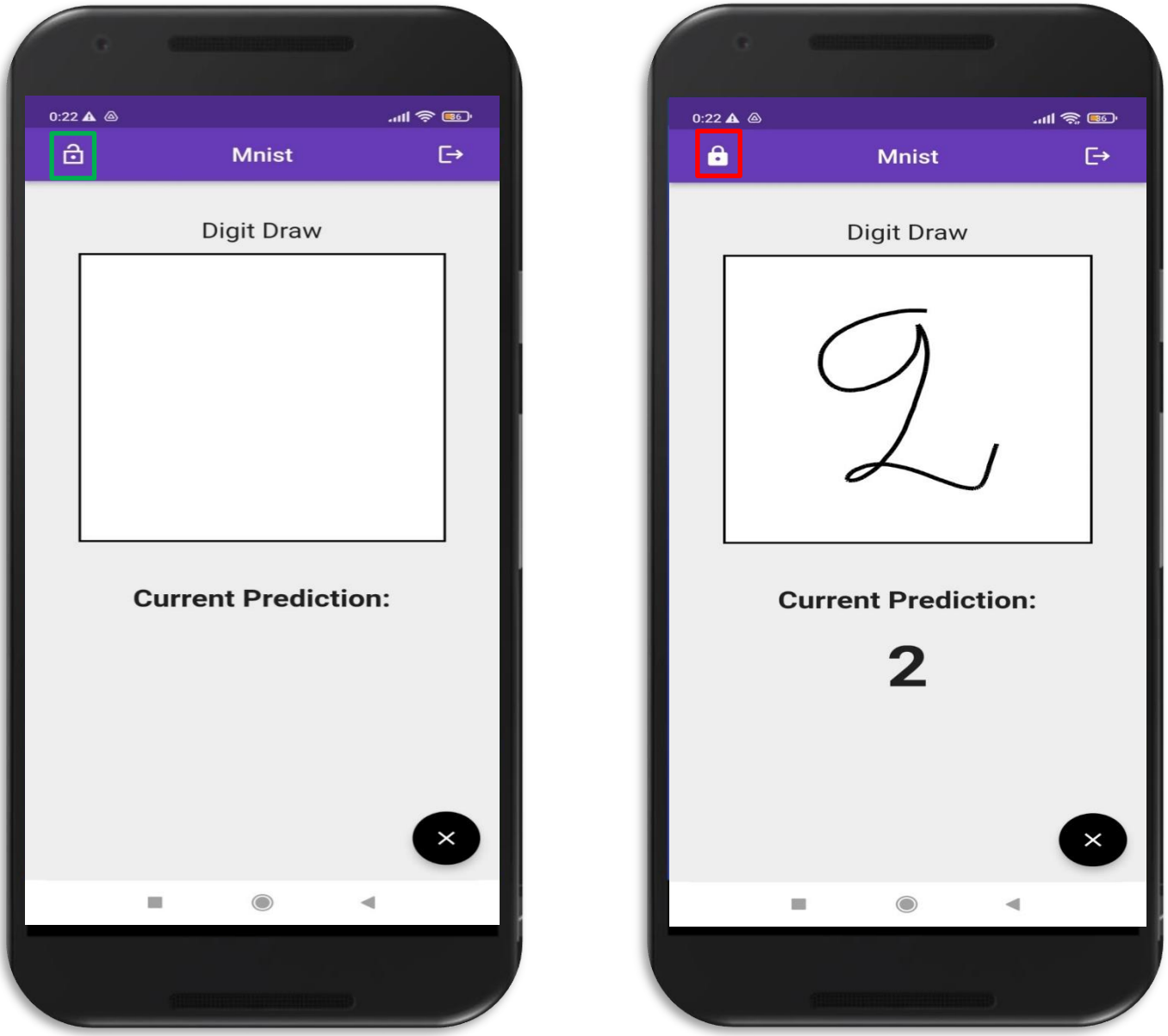


Figure 23 : résultat de prédiction

⇨ **La deuxième interface** : pour design un numéro et voir la production de notre model

Remarque :

Pour design un numéro, vous devez d'abord cliquer sur le Button en haut de la page « Verrouiller ».



Pour effacer que vous avez déjà écrit Vous devez cliquer sur le bouton en bas de la page (X)

2- L’application de la reconnaissance faciale

Première interface contiendra seulement deux Button :

- **Capter une photo :** pour prendre une photo avec une interface opencv.
- **Quitter :** qui nous permet de quitter l’application.

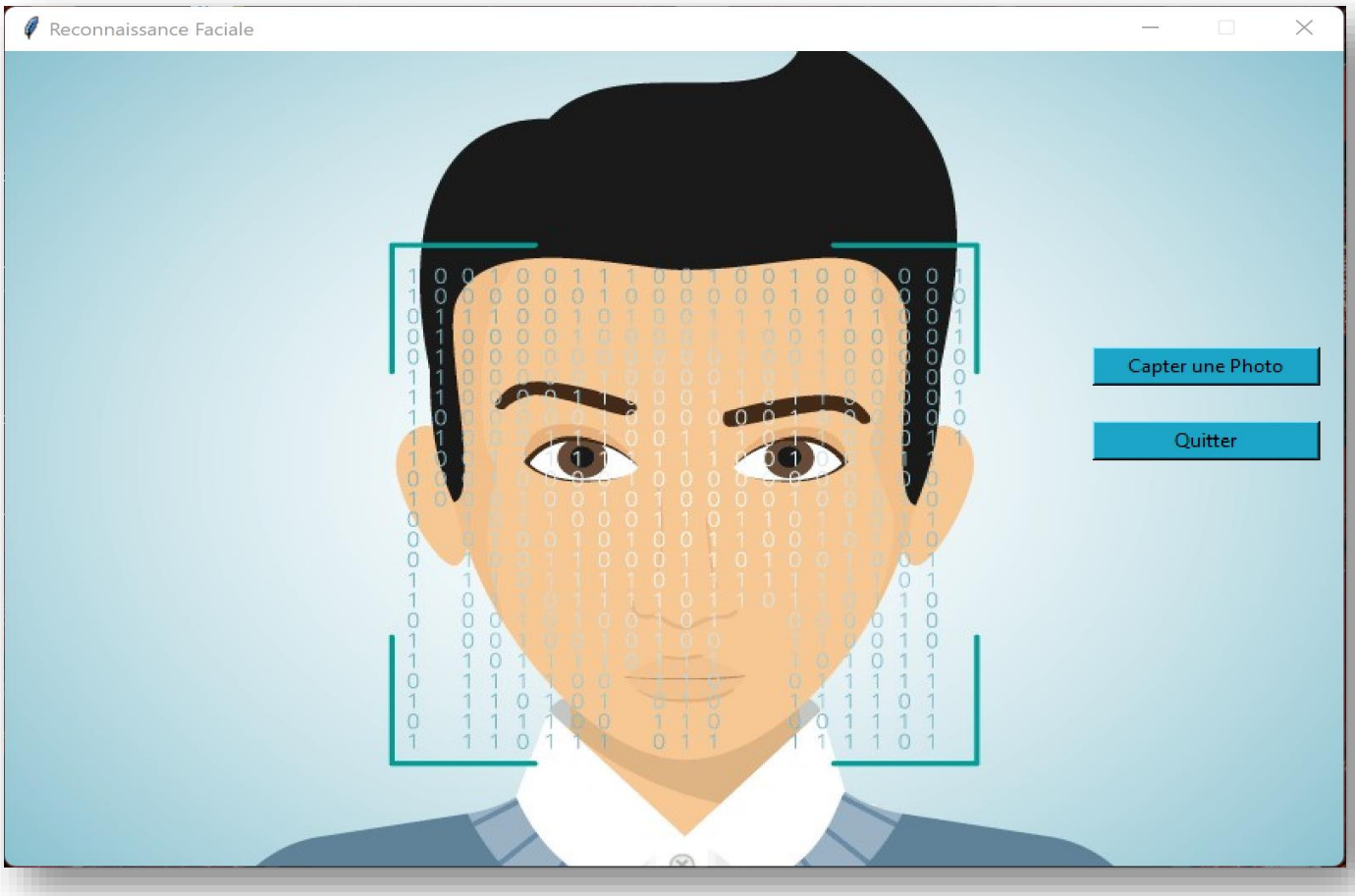


Figure 24 : L’interface D’accueil

Deuxième interface avec opencv pour prendre une photo, il y a deux événements a déclenché
Si on clique sur :

- **Espace :** nous permettons de capter le visage de quelqu’un afin de nous donnera à notre model afin d’afficher dans une autre interface la prédiction des 5 premières personnes similaires à lui.
- **Echap :** pour quitter l’application.

Figure 25 : Interface pour prendre une photo

Troisième interface pour afficher les cinq premières personnes similaires à l'image qui est prendre par l'interface précèdent :

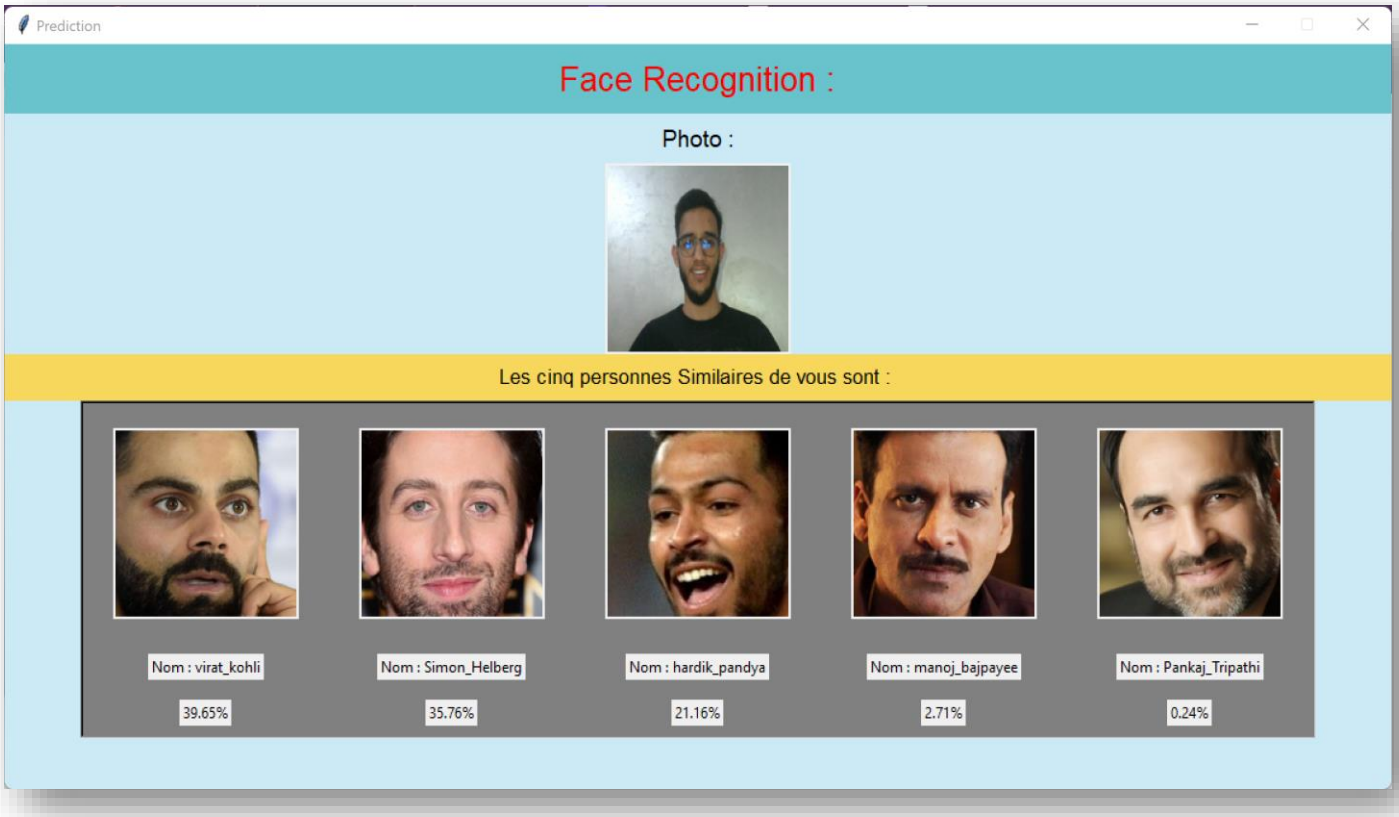


Figure 26 : Interface_1 d'affichage de 5 premières personnes

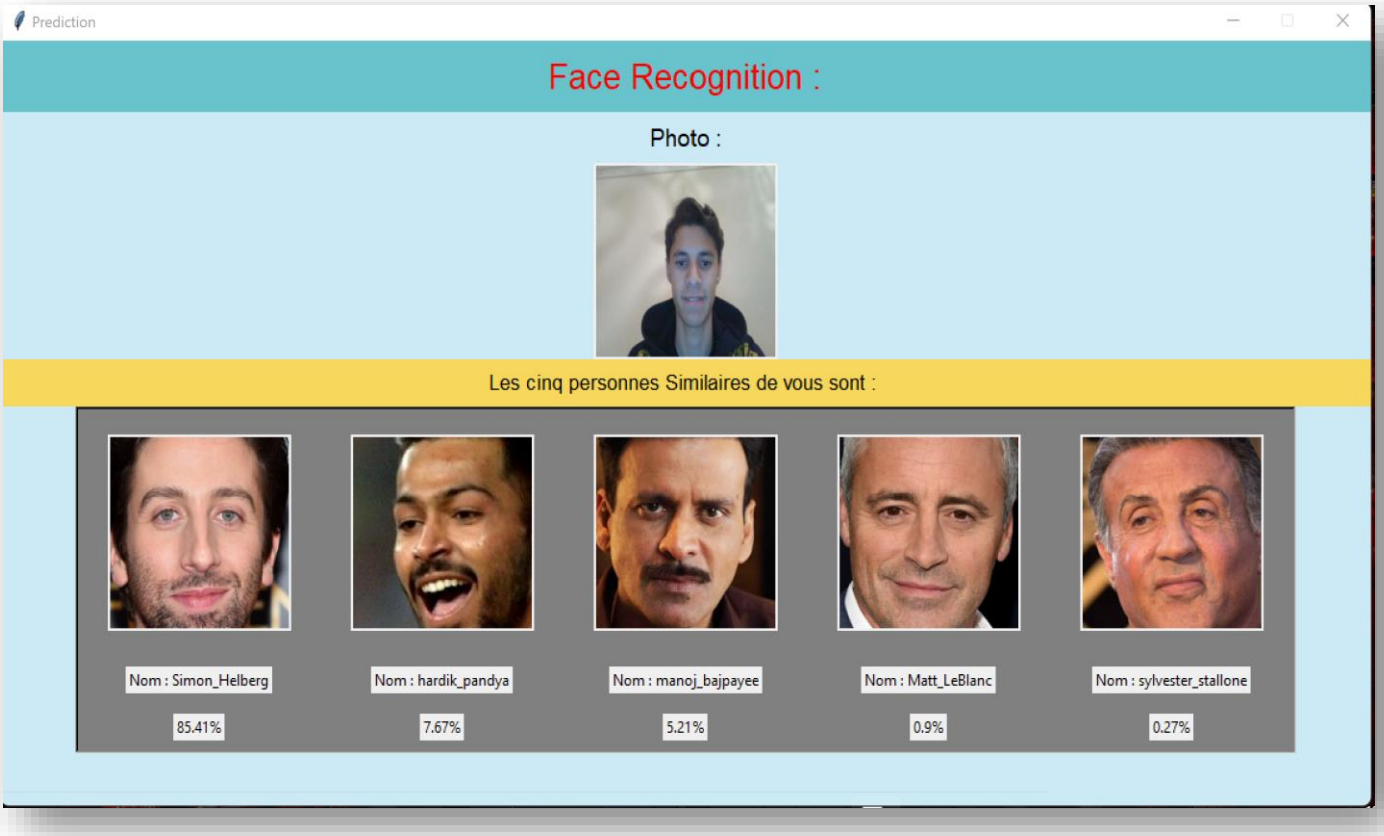


Figure 27 : Interface_2 d'affichage de 5 premières personnes

Conclusion

La réalisation de notre projet a été effectuée après trois étapes principales, la première, c'est la préparation de Dataset, la deuxième étape, c'est la construction du model pour qu'on puisse passer à la dernière étape qui est l'entraînement de model ainsi que le test.

Dans le cadre de notre projet, nous avons été permis de réaliser deux applications, une application mobile pour le projet MNIST et une application desktop pour notre projet de la reconnaissance faciale.

Tout au long de l'élaboration du projet, nous avons rencontré des difficultés tant au niveau de l'entrainement de notre model (il faut voir des machines performantes) qu'au niveau de la préparation de Dataset. Tout de même, nous avons réussi à les surpasse pour présenter en fin de compte des applications opérationnelles qui donnent des prédictions précises.

Webographie

https://flutter.dev/docslibrary.html

https://stackoverflow.com

https://www.journaldugeek.com/dossier/intelligence-artificielle-aujourdhui-fantasme-demain-revolution/

https://experiences.microsoft.fr/articles/intelligence-artificielle/comprendre-utiliser-intelligence-artificielle/

https://www.kaggle.com