MAI 2024

Kevin LE GRAND

CAS PRATIQUE : IA pour la reconnaissance d'iris

Développeur en intelligence artificielle

Certification RNCP34757

SIMPLON.CO

ISEN ALL IS DIGITAL! OUEST | yncréa

Microsoft

En adressant ce document à l'enseignante, je certifie que ce travail est le mien et que j'ai pris connaissance des règles relatives au référencement, au plagiat, ainsi qu'à l'usage d'une intelligence artificielle d'aide à la rédaction de type ChatGPT

Sommaire

1 Eléments du cas pratique

1.1 Mise en situation

Vous êtes un développeur IA, votre entreprise vous a confié la mission de développer une interface de reconnaissance d'oeil pour une entreprise souhaitant authentifier ses 45 employés à partir d'un scan de leurs yeux. Un employé avait déjà travaillé sur ce projet et vous livre la base de données, ainsi que son notebook qui avait permis la préparation de sa base de données et l'entrainement d'un modèle à l'authentification d'un employé à partir de l'image d'oeil gauche. Aussi, cet employé avait récupéré le modèle entrainé dans une application de prédiction qu'il avait développé en tkinter. Cependant, des problèmes subsistent : le modèle n'atteint pas les performances optimales, l'interface utilisateur manque d'ergonomie et se limite uniquement à la prédiction à partir des images d'yeux gauches. En outre, les prédictions actuelles fournissent les probabilités d'appartenance à chaque employé plutôt que l'identifiant respectif et les informations personnelles.

1.2 Fichiers fournis

- Jeu de Données :
  - Le jeu de données comprend 45 sous-dossiers, chacun correspondant à l'identifiant d'un employé (de 1 à 46, excluant 4).
  - Chaque sous-dossier contient des images d'oeil gauche et droit, réparties dans les répertoires "left" et "right".

- Un fichier JSON stockant les informations de chaque employé (pour chaque ID, il fournit : nom, date_embauche, genre, poste).

- Notebook d'Entraînement : Le notebook fournit les étapes détaillées de la préparation des données et de l'entraînement du modèle pour la reconnaissance d'oeil gauche.

- Application d'Authentification d'oeil gauche.

1.3 Cahier des charges

Améliorer le programme d'IA existant pour permettre la bonne classification d'un employé à partir d'un scan d'un de ses yeux (oeil droit ou oeil gauche, besoin d'un classifieur pour chaque).

- Développer une application conviviale qui réalise les tâches suivantes :
  - Permettre le téléchargement d'une image de l'oeil d'un employé.
  - Afficher l'image téléchargée.
  - Prédire si l'oeil est gauche ou droit.
  - Authentifier l'employé en utilisant le classifieur approprié en fonction de la prédiction précédente et en affichant les informations de la personne authentifiée.

2 Interprétation des indicateurs de performance de l'intelligence artificielle disponibles

2.1 Notebook

Les performances contenues dans le notebook du projet d'origine montrent une Accuracy de 76% sur l'œil gauche et que le model a uniquement été entraîné sur ce dernier. L'entraînement du modèle a été effectué sans augmentation de données et avec un petit nombre d'epoch. Il n'est pas facile de voir simplement où le modèle s'est trompé puisque la matrice de confusion est petite et que les résultats sont inscrits en code couleurs.

2.2 Application Tkinter

L'application ne permet pas de prédire la classification d'un employé pour les raisons suivantes :

- Il n'y a que le modèle pour l'œil gauche
- Le label encodeur n'a pas été importé donc la prédiction d'un employé en fonction de son œil gauche est erronée
- L'application ne permet pas le rapprochement et l'affichage entre la prédiction du modèle et le fichier json où sont contenu les informations des employés donc la vérification entre prédiction et réalité est impossible
- L'affichage de la probabilité de la prédiction par le modèle n'est pas réalisé dans le programme de l'application

3 Définition des caractéristiques des améliorations à apporter

3.1 Répartition des données

Pour que l'application corresponde au cahier des charges il va falloir créer 2 autres modèles, un pour l'œil droit et un pour prédire si c'est un œil droit ou gauche. Pour réaliser cette étape par créer 6 listes :

- Une paire de liste pour l'œil droit (X_right, Y_right)
- Une autre paire pour l'œil gauche (X_left, Y_left)
- Une dernière paire avec l'œil gauche et droit (X_r_or_l, Y_r_or_l)

Les listes X_right et X_left sont composées des images redimensionnées et standardisées en objet np.ndarray. Les listes Y_right et Y_left contiennent l'ID de chaque employé.

La liste X_r_or_l contient toutes les images et ont le même preprocessing que pour X_right et X_left. Y_r_ou_l est associé à chaque image par 0 ou 1, 0 correspondant à un œil droit et 1 l'œil gauche.

3.2 Augmentation des données

Pour améliorer les performances du modèle, la première solution est de lui fournir plus de données en augmentant le nombre d'image en entrée. L'augmentation de données se fera à l'aide de la classe ImageDataGenerator de Tensorflow. Une fonction sera réalisée afin de pouvoir choisir le nombre de fois que l'on désire augmenter une image. Les transformations ajoutées pourront être la rotation, la luminosité ou le zoom.

Pour effectuer cette transformation les images seront prises une à une, on inversera la standardisation, on effectuera la transformation, puis on standardisera la nouvelle image.

3.3 Améliorations diverse

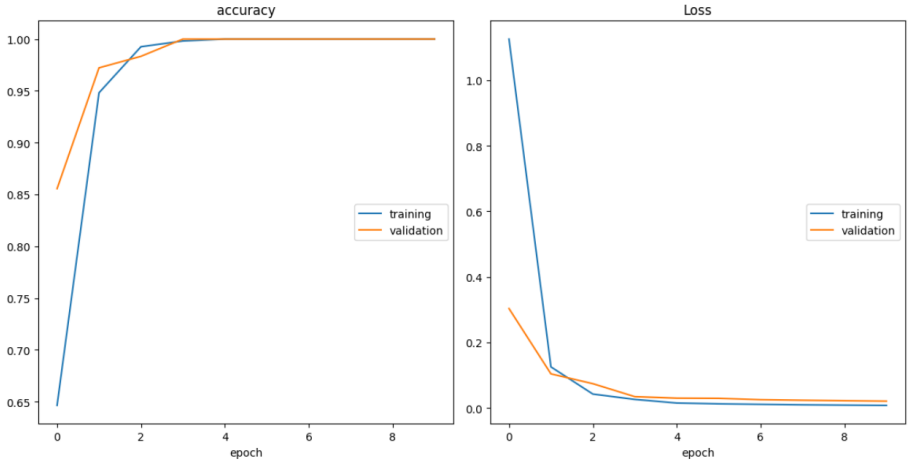
Les autres améliorations possibles seront :

- Augmenter le nombre d'epochs
- Régler les paramètres des callbacks ReduceLROnPlateau et Earlystopping afin d'optimiser la descente de gradient pour ajuster au mieux les performances et d'éviter le surentrainement du modèle
- Prévoir la sauvegarde d'image des données tests pour l'application
- Créer une fonction pour l'affichage des performances et de la matrice de confusion afin que les résultats soient plus facilement interprétables
- Sauvegarder des images test avec son numéro d'identifiant dans le nom du fichier afin de permettre la vérification dans l'application

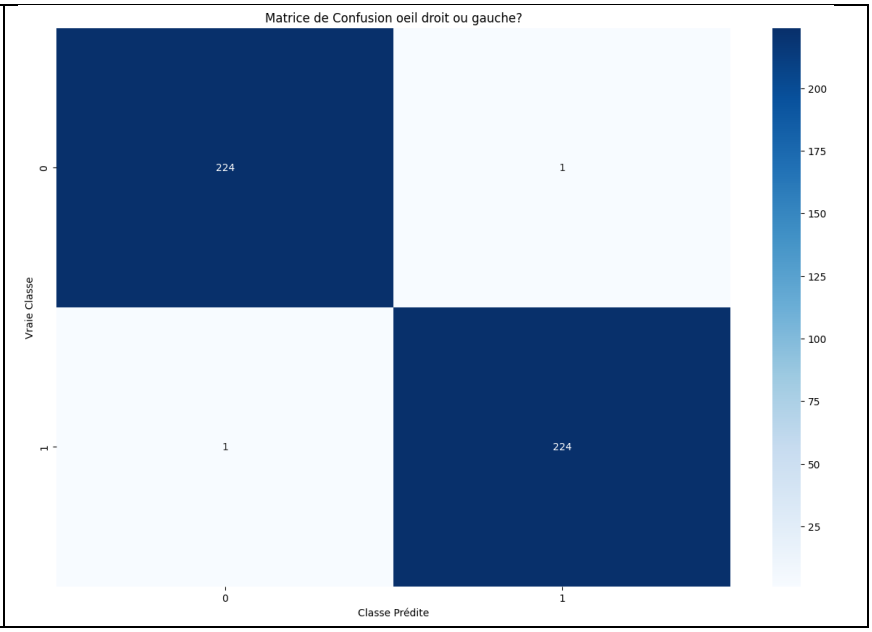
4 Intégration des améliorations à l'algorithme d'intelligence artificielle

Après avoir effectué ces améliorations les résultats de l'algorithme d'intelligence artificielle sont les suivants :

4.1 Prédiction œil droit ou gauche

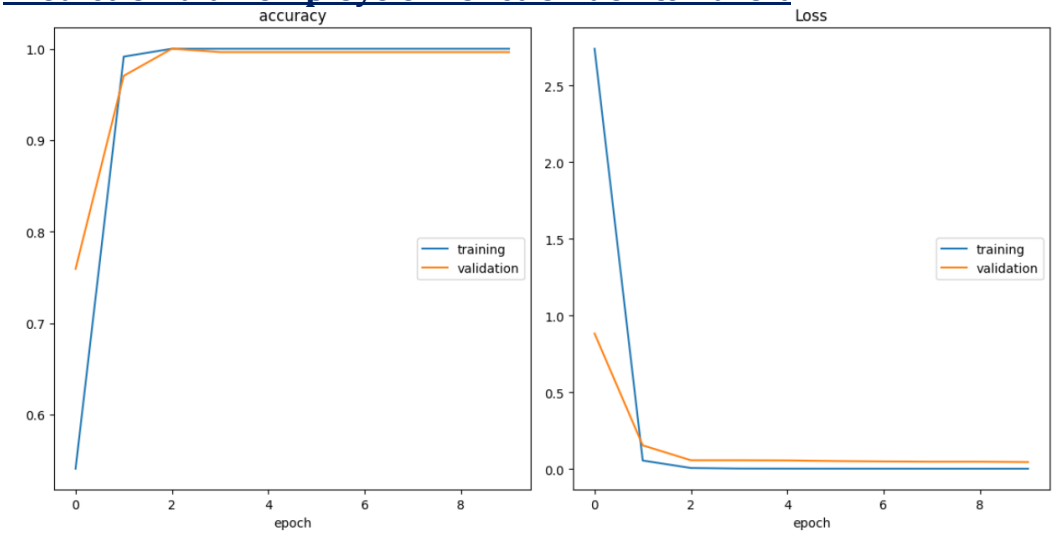



```
Précision : 1.00
Rappel : 1.00
Score F1 : 1.00
Exactitude : 1.00
```

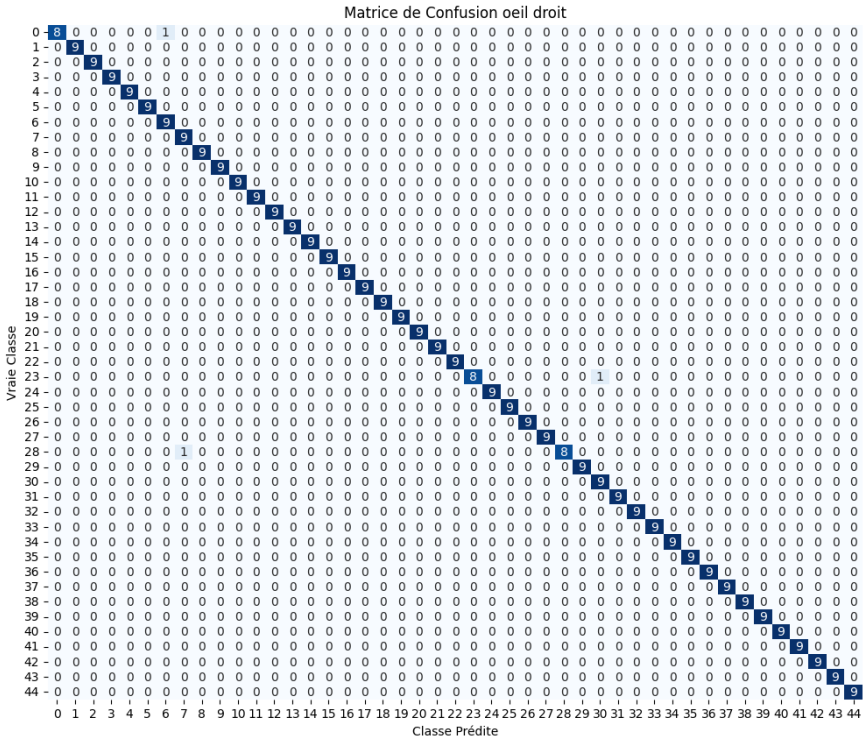


Sur les 450 prédictions à réaliser le modèle ne s'est trompé que deux fois, une fois sur chaque œil.

4.2 Prédiction d'un employé en fonction de l'œil droit




```
Précision : 0.99
Rappel : 0.99
Score F1 : 0.99
Exactitude : 0.99
```



On remarque que le modèle s'ajuste très rapidement et qu'il atteint des performances presque maximums au bout de quelques epochs. Sur les 405 prédictions à réaliser le modèle ne s'est trompés que 3 fois ce qui reste acceptable par rapport au modèle d'origine.

4.3 Prédiction d'un employé en fonction de l'œil gauche

Sur l'œil gauche les performances sont quasiment similaires à celles de l'œil droit avec une seule erreur sur l'ensemble des prédictions à réaliser.

5 Estimation de charge au regard du besoin d'évolution de l'application

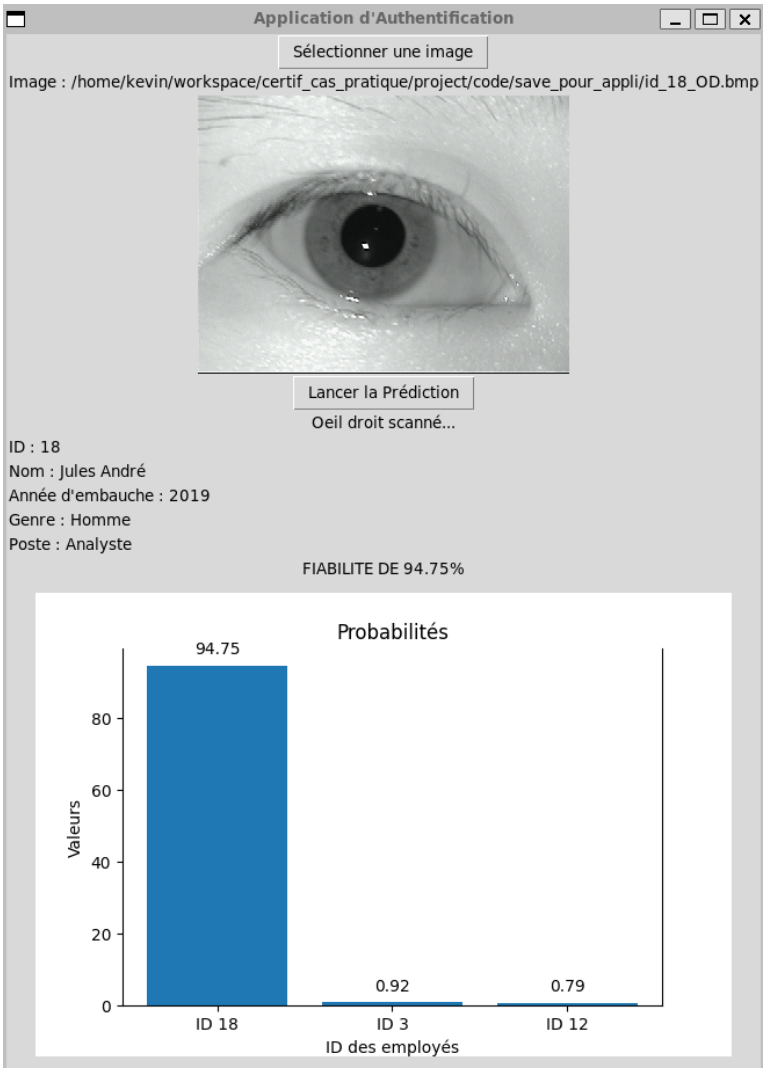
Afin d'avoir une application fonctionnelle et pertinente il va falloir apporter les modifications suivantes :

- Importer les label encodeur afin de déterminer l'ID de l'employé prédit
- Importer les 3 modèles permettant de détecter un employé en fonction de son œil droit ou gauche
- Créer un algorithme permettant de détecter en premier si c'est un œil droit ou gauche puis d'utiliser le modèle correspondant pour réaliser la prédiction
- Utiliser le label encodeur pour déterminer l'ID de l'employé
- Faire le rapprochement avec la liste des employés contenus dans le fichier json
- Afficher l'œil détecté ainsi que les caractéristiques de l'employés
- Afficher la fiabilité de la prédiction

6 Intégration de l'évolution fonctionnelle

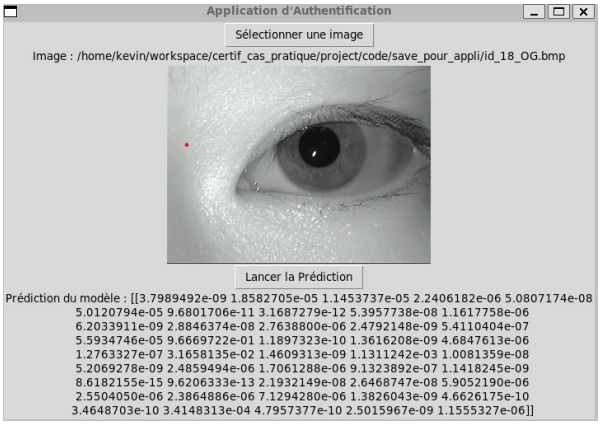
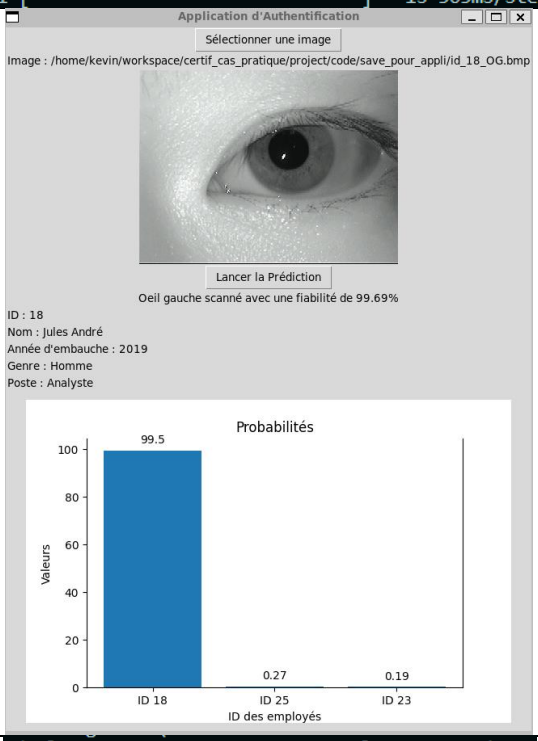
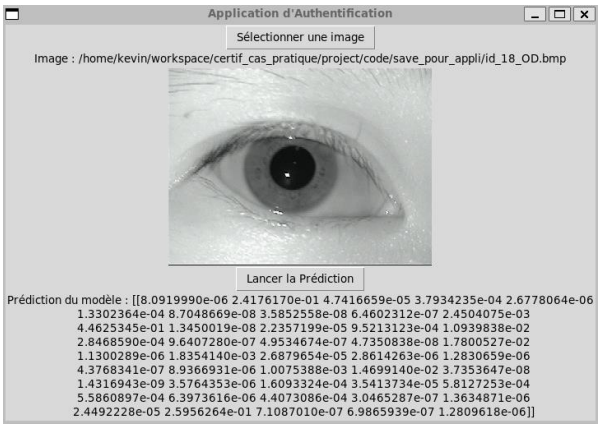
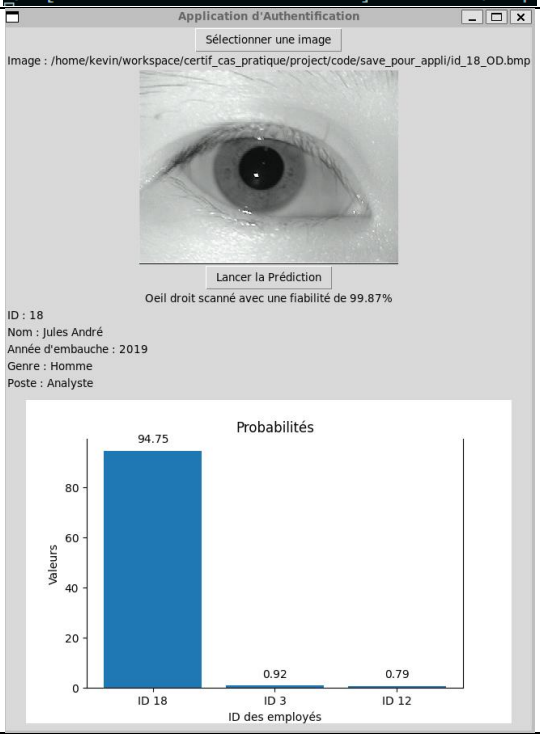
Une fois ces intégrations réalisées l'application ressemble à ceci :

- L'ID est bien présent dans le nom de l'image à prédire
- L'indication du type d'œil est indiquée
- Les caractéristiques de l'employé prédit sont affichées
- L'application permet également de voir les 3 identifiants les plus probables en fonction de l'œil fourni



7 Test de la non-régression de l'application à la suite de l'intégration de l'évolution

7.1 Comparatif des performances et visualisation

		Appli d'origine	Appli évoluée
Œil gauche	Temps	`1/1 [==============================] - 1s 611ms/step`	`1/1 [==============================] - 0s 432ms/step` `1/1 [==============================] - 1s 505ms/step`
	Affichage		
Œil droit	Temps	`1/1 [==============================] - 0s 374ms/step`	`1/1 [==============================] - 1s 560ms/step` `1/1 [==============================] - 1s 572ms/step`
	Affichage		

7.2 Conclusion sur la non-régression

Le temps d'exécution sur la version évoluée de l'application est moins performant, une demie seconde pour l'application d'origine contre plus d'une seconde pour la version évoluée. Ceci est dû au fait que l'application évoluée doit faire 2 prédictions au lieu d'une seule, une prédiction pour définir si c'est un œil droit ou gauche puis prédire l'ID de l'employé.

Concernant les performances de prédictions il est difficile de faire un comparatif. Néanmoins la probabilité la plus élevée avec l'œil gauche pour l'application d'origine est d'environ 96,6% contre 99,5% dans l'application évoluée. De plus, n'ayant pas le label encodeur d'importer dans l'application d'origine, il est impossible de savoir si cette probabilité de 96,6% correspond bien à l'ID 18.

Pour finir en termes d'affichage l'évolution est bien présente puisqu'il est facile de savoir quel employé a été détecté et quel est la fiabilité de cette prédiction.

8 Liens GitHub du projet

https://github.com/rastakoer/certif_CasPratique