

Nom&Prénom: BOUTALBI Mohammed Iliass

Groupe: 01

Master 2 IA&IOT

**Rapport du projet Deep Learning:
“Reconnaissance d’empreinte digitale”**

Intitulé :

Développement d'un modèle de Deep Learning pour la reconnaissance d'empreintes digitales

Description :

Mise en place d'un système de reconnaissance d'empreintes digitales en utilisant des réseaux de neurones convolutifs (CNN). Le projet inclut la collecte et le prétraitement des images d'empreintes, l'entraînement d'un modèle de classification sur un dataset de référence, ainsi que l'évaluation des performances du modèle en termes de précision.

Principe:

Ce processus permet au modèle d'apprendre à partir des images du dataset SOCOFing et d'appliquer les règles afin de créer le modèle et prédire, de façon précise, la valeur de sortie lorsqu'une valeur d'entrée est donnée.

Prédire les résultats en fonction d'un ensemble d'entrées(images d'empreinte).

Utilisation:

Dataset utilisée:

<https://www.kaggle.com/api/v1/datasets/download/ruizgara/socofing>

Apprentissage supervisé

Par exemple, elle peut apprendre à reconnaître une empreinte digitale après qu'on lui ait montré des milliers de photos d'empreinte.

Application: vision par ordinateur.

Les couches:

input layer: les images d'empreinte digitale(Real , Altered)

hidden layer: calcul mathématiques

output layer: prédire l'empreinte digitale soit real ou altered

Outils :

- **Langage :** Python
- **Frameworks :** TensorFlow, Keras
- **Bibliothèques :** OpenCV (pour le traitement d'image), NumPy, Matplotlib
- **Environnement :** Pycharm

Comment fonctionnent les CNN dans la reconnaissance d'empreintes digitales :

- Les réseaux de neurones convolutifs (CNN) sont des réseaux neuronaux spécialisés conçus pour traiter des données structurées en grille, comme les images. Dans la reconnaissance d'empreintes digitales, les CNN peuvent être utilisés pour extraire des caractéristiques distinctives des images d'empreintes et les classer comme correspondantes ou non correspondantes.

Aperçu du projet avec des CNN (Convolutional Neural Networks):

Prétraitement des données :

Comme pour tout projet en deep learning, vous devrez prétraiter les images d'empreintes digitales. Cela comprend :

Redimensionnement des images : Adapter toutes les images à une taille uniforme.

Normalisation des valeurs de pixels : Ajuster les valeurs des pixels pour améliorer la performance et la stabilité du modèle.

Augmentation des données : Appliquer des transformations comme la rotation ou le redimensionnement pour diversifier les exemples d'entraînement et améliorer la robustesse du modèle.

Architecture CNN :

Couches Convolutionnelles : Extraire des caractéristiques telles que les contours et textures des empreintes digitales.

Couches de Pooling : Réduire la dimensionnalité en sous-échantillonnant l'image, ce qui permet de diminuer la quantité de calcul nécessaire tout en conservant les caractéristiques essentielles.

Couches Entièrement Connectées : Après l'extraction de caractéristiques, ces couches déterminent si l'empreinte digitale appartient à la classe 'Real' ou 'Altered' et si la personne existe ou non.

Implementation Steps:

- **Dataset:** SOCOFing
- **Outils:**
 - **Python**
 - **TensorFlow/Keras** pour construire le modèle CNN.
 - **OpenCV** pour le traitement des images.
 - **NumPy** pour la gestion des données.

Quand on exécute le code 'main.py' il va nous créer le modèle "fingerprint_cnn_model.keras" et après on va créer un fichier 'predict.py' pour charger le modèle et tester notre modèle.

L'exécution du code 'main.py' (avec seulement 5000 images réelles et 7000 altérées juste pour l'exécution ne prend pas beaucoup de temps)

Project: finger

Files: fingerprint_cnn_model.h5, fingerprint_cnn_model.keras, fingerprint_cnn_model1.keras, main.py, predict.py, tes.py

```
25 def load_images_from_folder(folder, label, limit=None): 2 usages
35     images.append(img)
36     labels.append(label)
37     return images, labels
38
39 # Charger 150 images du dossier "Real"
40 real_images, real_labels = load_images_from_folder(real_path, label="Real", limit=5000) # Limite d'images
41
42 # Charger toutes les images du dossier "Altered"
43 altered_images = []
44 altered_labels = []
45 for altered_type in ["Altered-Easy", "Altered-Medium", "Altered-Hard"]:
46     path = os.path.join(altered_path, altered_type)
47     imgs, lbls = load_images_from_folder(path, label="Altered", limit=7000) # Pas de limite ici
48     altered_images.extend(imgs)
49     altered_labels.extend(lbls)
50
```

Run: main

Epoch 1/4
325/325 1268s 4s/step - accuracy: 0.8006 - loss: 0.5221 - val_accuracy: 0.8504 - val_loss: 0.3212

Epoch 2/4
325/325 18s 54ms/step - accuracy: 0.8692 - loss: 0.2906 - val_accuracy: 0.8473 - val_loss: 0.3265

Epoch 3/4
325/325 18s 54ms/step - accuracy: 0.8890 - loss: 0.2443 - val_accuracy: 0.8425 - val_loss: 0.3281

Epoch 4/4

Project: finger

Files: fingerprint_cnn_model.h5, fingerprint_cnn_model.keras, fingerprint_cnn_model1.keras, main.py, predict.py, tes.py

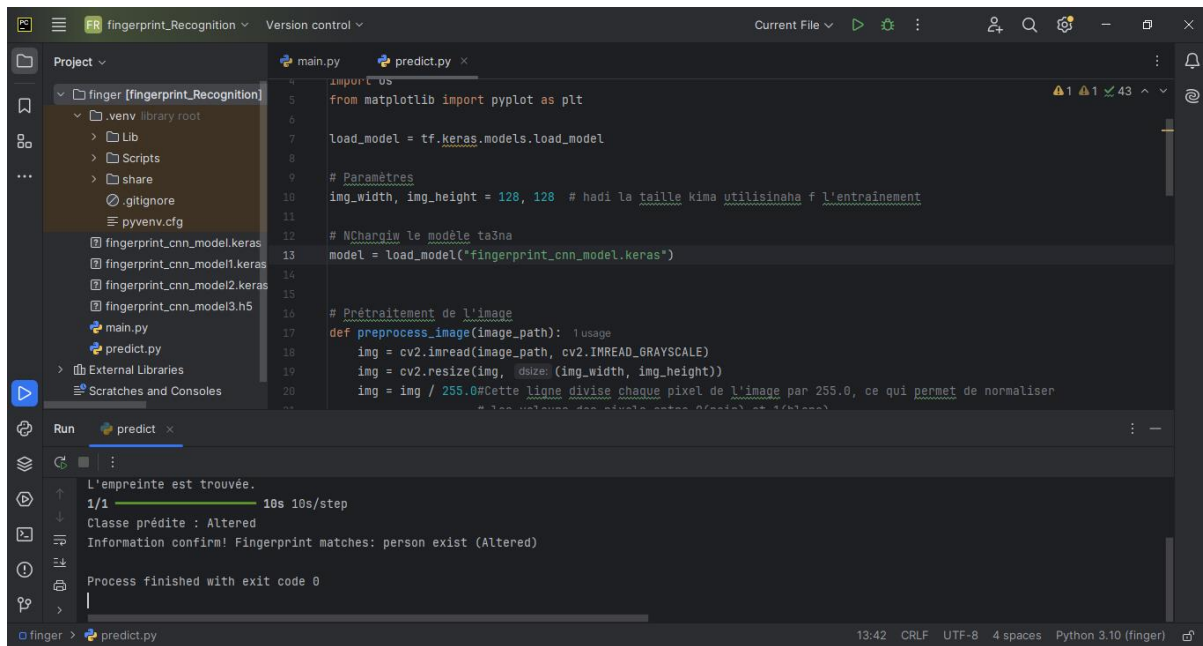
```
25 def load_images_from_folder(folder, label, limit=None): 2 usages
35     images.append(img)
36     labels.append(label)
37     return images, labels
38
39 # Charger 150 images du dossier "Real"
40 real_images, real_labels = load_images_from_folder(real_path, label="Real", limit=5000) # Limite d'images
41
42 # Charger toutes les images du dossier "Altered"
43 altered_images = []
44 altered_labels = []
45 for altered_type in ["Altered-Easy", "Altered-Medium", "Altered-Hard"]:
46     path = os.path.join(altered_path, altered_type)
47     imgs, lbls = load_images_from_folder(path, label="Altered", limit=7000) # Pas de limite ici
48     altered_images.extend(imgs)
49     altered_labels.extend(lbls)
50
```

Run: main

Epoch 4/4
325/325 21s 64ms/step - accuracy: 0.9007 - loss: 0.2277 - val_accuracy: 0.8648 - val_loss: 0.2809
163/163 2s 10ms/step - accuracy: 0.8645 - loss: 0.2705
Test Loss: 0.28088390827178955, Test Accuracy: 0.864807665348053

Process finished with exit code 0

L'exécution du code 'predict.py':



The screenshot shows a Jupyter Notebook interface with a dark theme. The left sidebar displays the project structure for 'finger [fingerprint_recognition]', including files like 'fingerprint_cnn_model.keras', 'main.py', and 'predict.py'. The main area shows the code for 'predict.py', which imports 'tf.keras.models' and 'matplotlib.pyplot', loads a model, and processes an image. The bottom panel shows the output of the code execution, indicating that the fingerprint was found and the predicted class is 'Altered'.

```
import os
from matplotlib import pyplot as plt

load_model = tf.keras.models.load_model

# Paramètres
img_width, img_height = 128, 128 # hadi la taille kima utilisinaha f l'entrainement

# NChargiw le modèle ta3na
model = load_model("fingerprint_cnn_model.keras")

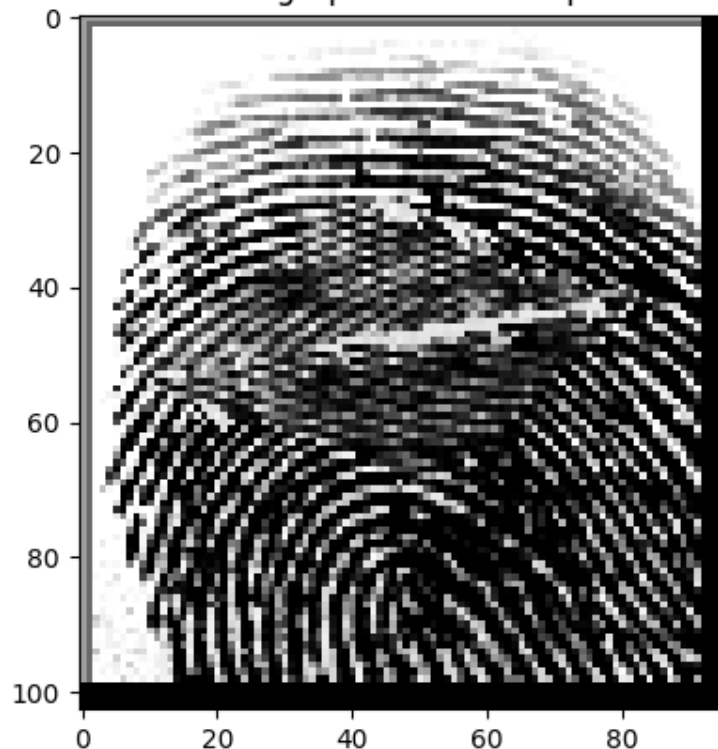
# Prétraitement de l'image
def preprocess_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (img_width, img_height))
    img = img / 255.0 # Cette ligne divise chaque pixel de l'image par 255.0, ce qui permet de normaliser
    # les valeurs des pixels entre 0 (noir) et 1 (blanc)
```

Run predict

L'empreinte est trouvée.
1/1 10s 10s/step
Classe prédite : Altered
Information confirm! Fingerprint matches: person exist (Altered)
Process finished with exit code 0

Figure 1

Information confirm! Fingerprint matches: person exist (Altered)



Note: Vous trouverez en piece jointe les fichiers code dans l'email

