

Rapport du Projet : Hand Gesture Classification avec une interface GUI Java en utilisant les method RMI et Socket

Introduction

Ce projet vise à classifier les gestes de la main en utilisant un modèle de Deep Learning basé sur **CNN (Convolutional Neural Network)**. L'architecture repose sur plusieurs technologies :

- **Un modèle de classification d'images**
 - **Un serveur Flask** pour héberger l'API du modèle
 - **Une communication entre Java et Flask via Sockets et RMI**
 - **Une interface graphique en Swing** pour interagir avec le système
-

1. Modèle de Classification des Gestes

Le modèle utilisé est basé sur une architecture **CNN** entraînée pour reconnaître différentes positions de la main.

Détails techniques :

- **Type de modèle** : CNN (Convolutional Neural Network)
- **Framework utilisé** : TensorFlow / Keras
- **Base de données** : Dataset contenant des images de gestes de la main
- **Prétraitement des images** : Redimensionnement, normalisation
- **Architecture** :
 - Convolutional Layers
 - Max Pooling Layers
 - Fully Connected Layers
 - Softmax pour la classification

Code d'entraînement du modèle :

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

mod=models.Sequential()

# CNN Layer 1

mod.add(layers.Conv2D(64,(3,3),input_shape=(28,28,1),padding='same'))
mod.add(layers.BatchNormalization())
mod.add(layers.Activation('relu'))

# CNN Layer 2
```

```
mod.add(layers.Conv2D(128,(3,3),padding='same'))
mod.add(layers.BatchNormalization())
mod.add(layers.Activation('relu'))

# CNN Layer 3
mod.add(layers.Conv2D(256,(3,3),padding='same'))
mod.add(layers.BatchNormalization())
mod.add(layers.Activation('relu'))
mod.add(layers.MaxPooling2D())
mod.add(layers.Dropout(0.2))

# CNN Layer 4

mod.add(layers.Conv2D(50,(3,3),padding='same'))
mod.add(layers.BatchNormalization())
mod.add(layers.Activation('relu'))
mod.add(layers.MaxPooling2D())
mod.add(layers.Dropout(0.2))

# CNN Layer 5

mod.add(layers.Conv2D(25,(3,3),padding='same'))
mod.add(layers.BatchNormalization())
mod.add(layers.Activation('relu'))
mod.add(layers.Dropout(0.1))

# Flattening

mod.add(layers.Flatten())

# Fully Connected layer 1
mod.add(layers.Dense(1024,activation='relu'))
mod.add(layers.BatchNormalization())

# Fully Connected layer 2

mod.add(layers.Dense(512,activation='relu'))
mod.add(layers.BatchNormalization())

# Fully Connected layer 3

mod.add(layers.Dense(256,activation='relu'))
mod.add(layers.BatchNormalization())

# Fully Connected layer 4

mod.add(layers.Dense(64,activation='relu'))
mod.add(layers.BatchNormalization())
mod.add(layers.Dropout(0.2))

# Output Layer
```

```
mod.add(layers.Dense(24, activation='softmax'))
```

2. Serveur Flask

Le serveur Flask sert d'interface entre le modèle et les applications clientes.

Fonctionnalités

- **Recevoir une image en POST**
- **Effectuer la prédiction**
- **Retourner le résultat au client**
- **Utilisation de ThreadPoolExecutor pour gérer les requêtes en parallèle.**
- **Ajout de threaded=True pour activer le multi-threading dans Flask.**

```
@app.route('/', methods=['GET', 'POST'])
def home():
    print(f"Processing on Thread ID: {threading.get_ident()}")
    with open('log.txt', 'a') as f:
        f.write(f"Processing on Thread ID: {threading.get_ident()}\n")

    if request.method == 'POST':
        file = request.files.get('file')
        if file:
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
            file.save(file_path)

            predicted_alphabet = classify_image(file_path)

    return predicted_alphabet

return 'Please send a POST request with an image file.'
```

3. Communication entre Java et Flask

La communication entre Java et Flask est assurée par **Java Sockets** et **Java RMI**.

Architecture

- **Socket Server** : Reçoit les images depuis le client et les transfère au serveur Flask
- **Socket Client** : Envoie une image et attend une réponse
- **RMI Server** : Fournit une interface distante pour la classification
- **RMI Client** : Envoie des requêtes à RMI Server

Code du Socket Server en Java

```
// Source code is decompiled from a .class file using FernFlower decompiler.
import java.io.IOException;
```

```

import java.net.ServerSocket;
import java.net.Socket;

public class SocketServer {
    private static final int PORT = 5001;
    private static final String FLASK_SERVER_URL = "http://127.0.0.1:5000/";

    public SocketServer() {
    }

    public static void main(String[] var0) {
        try {
            ServerSocket var1 = new ServerSocket(5001);

            try {
                System.out.println("Serveur Java en ecoute sur le port 5001...");

                while(true) {
                    Socket var2 = var1.accept();
                    System.out.println("Nouvelle connexion reçue de " +
var2.getInetAddress());
                    (new Thread(new SocketServer$ClientHandler(var2))).start();
                }
            } catch (Throwable var5) {
                try {
                    var1.close();
                } catch (Throwable var4) {
                    var5.addSuppressed(var4);
                }

                throw var5;
            }
        } catch (IOException var6) {
            var6.printStackTrace();
        }
    }
}

```

Code du RMI Server en Java

```

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class RmiServer {
    public static void main(String[] args) {
        try {
            // Demarrer le registre RMI
            Registry registry = LocateRegistry.createRegistry(1099);

            // Instancier l'objet distant
            ClassifierServiceImpl service = new ClassifierServiceImpl();

```

```

    // Publier l'objet dans le registre
    registry.rebind("ClassifierService", service);

    System.out.println("Serveur RMI demarre...");
} catch (Exception e) {
    e.printStackTrace();
}
}

}
}

```

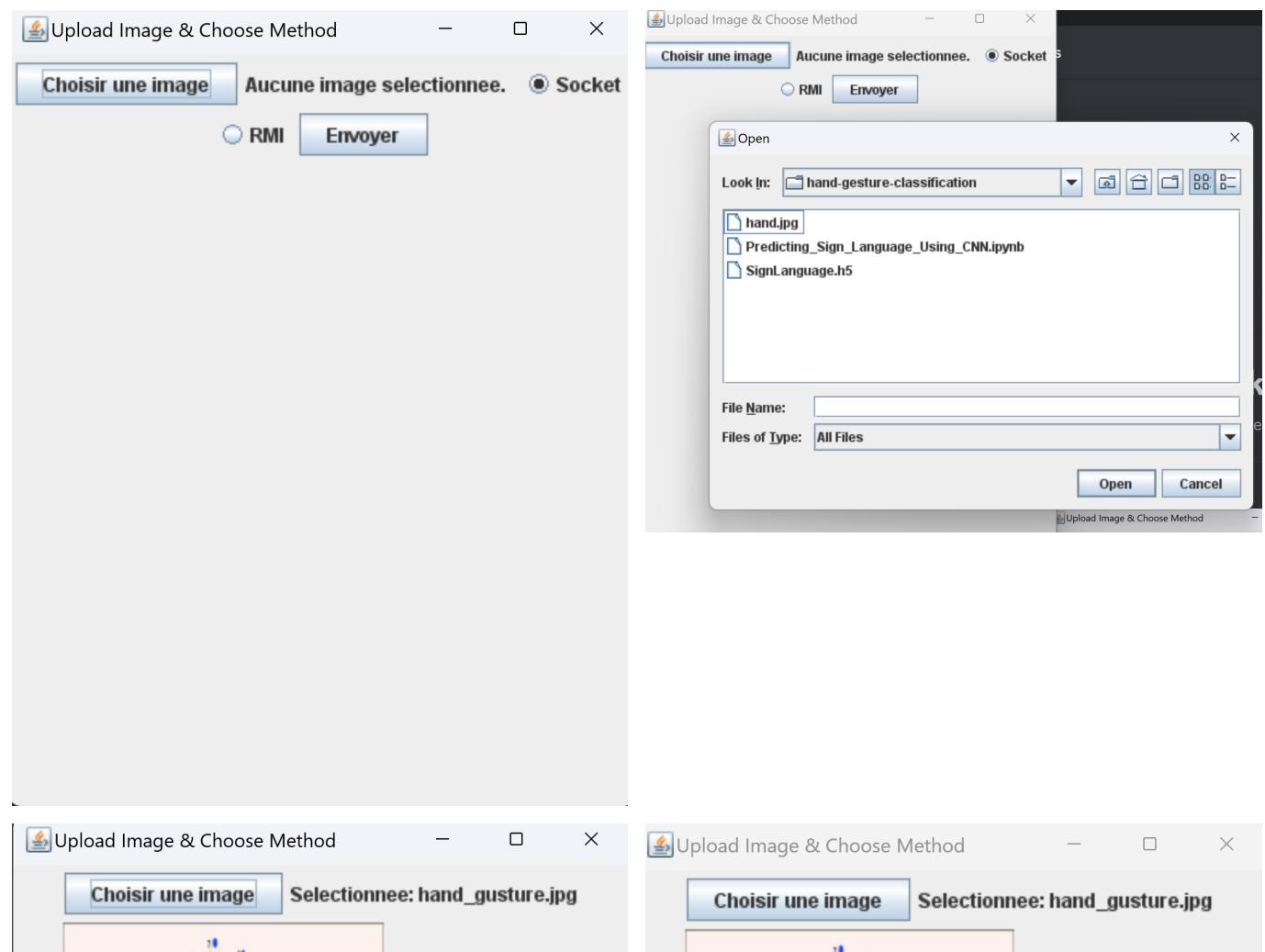
4. Interface Graphique avec Swing

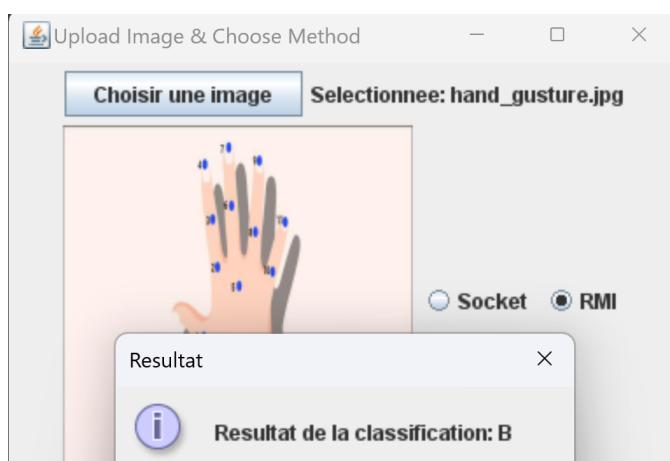
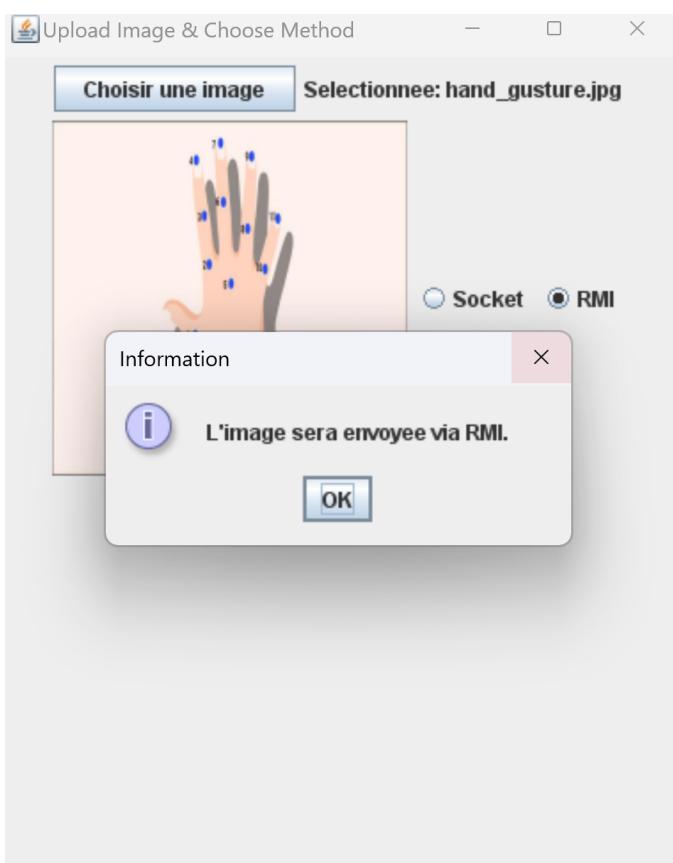
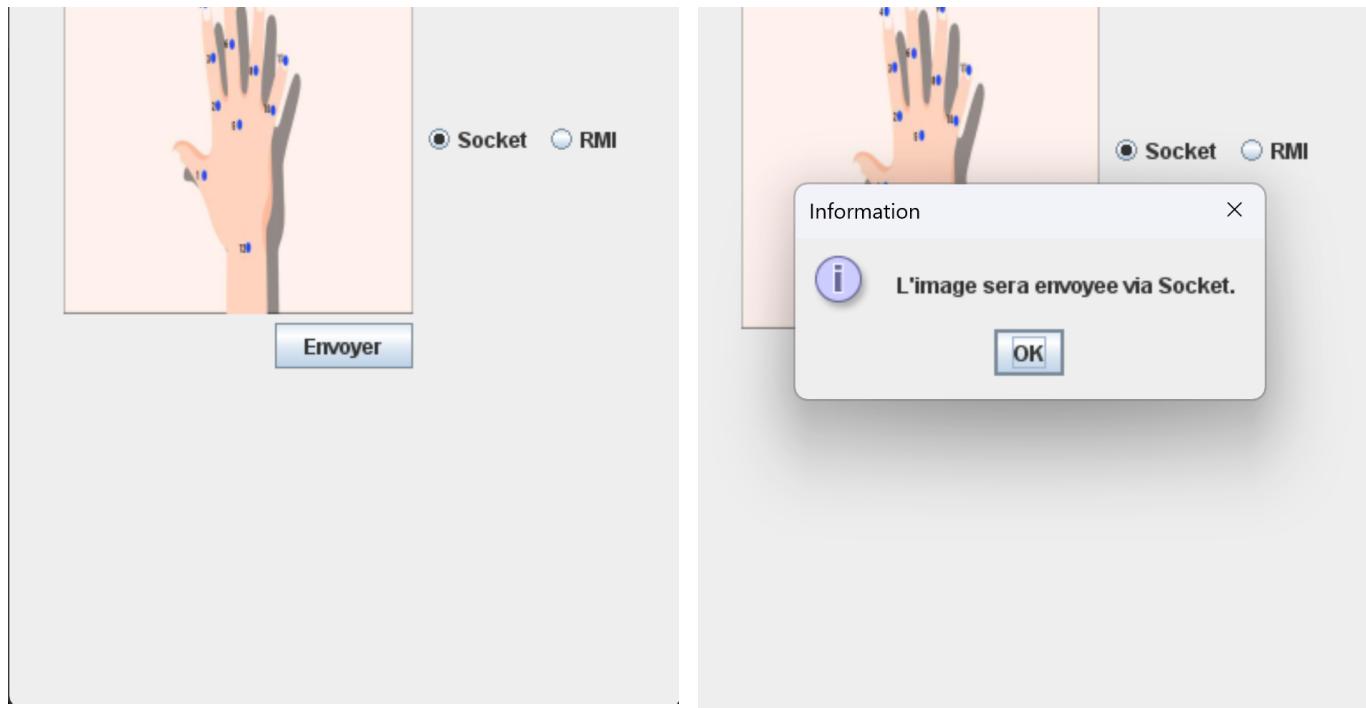
Une interface Swing a été développée pour permettre à l'utilisateur d'envoyer des images et d'afficher les prédictions.

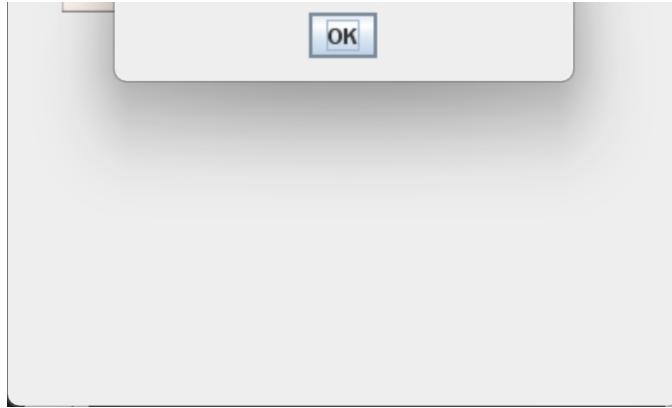
Fonctionnalités

- Bouton pour sélectionner une image
- Bouton pour envoyer l'image au serveur Flask
- Affichage du résultat de la classification

Démonstration







 **Projet réalisé par :**

- Kemmoun Ramzy
- Abadli Badreddine
- Khouas Assia
- Tareb Selma