



**INP Polytech Clermont Ferrand**

---

**Rapport du Projet d'Imagerie  
Numérique**

---

**Traducteur automatique de langage des  
signes**

---

**Effectué par**

Mustapha Lahmer  
Said Akhssay

**Encadrant :**

M. Omar AIT AIDER

02/02/2025

# Remerciements

Nous souhaitons exprimer notre profonde gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réalisation de ce travail.

Nous adressons tout d'abord nos sincères remerciements à **M. Omar AIT AIDER**, notre encadrant, pour son accompagnement précieux, ses conseils avisés et sa disponibilité tout au long de ce projet. Son expertise et son soutien nous ont été d'une grande aide dans l'aboutissement de ce travail.

Nos remerciements vont également à l'ensemble du corps professoral qui nous a transmis des connaissances essentielles et nous a guidés tout au long de notre parcours académique.

Enfin, nous tenons à exprimer notre reconnaissance à nos familles et amis pour leur soutien inconditionnel, leurs encouragements et leur patience, qui ont été une source de motivation précieuse durant cette aventure.

## Résumé

Ce projet vise à concevoir un système de traduction automatique du langage des signes en texte en temps réel en utilisant des techniques avancées de vision par ordinateur et d'apprentissage profond. En exploitant OpenCV et Mediapipe pour la détection des mains, ainsi que MobileNetV2 pour la classification des gestes, nous avons développé un modèle capable d'identifier avec précision plusieurs signes de la langue des signes. L'entraînement a été réalisé sur un dataset enrichi grâce à la technique de data augmentation afin d'améliorer la robustesse du modèle. Une interface interactive a également été mise en place pour permettre une détection fluide et rapide des signes via une webcam. Les résultats obtenus montrent une précision élevée, avec un taux de reconnaissance supérieur à 90 %. Ce système constitue une avancée significative en matière d'accessibilité et pourrait être étendu pour reconnaître un plus grand nombre de signes et améliorer la fluidité de la traduction.

**Mots-clés :** Langage des signes, Reconnaissance des gestes, Vision par ordinateur, Deep Learning, Traduction automatique.

## Abstract

This project aims to develop an automatic sign language translation system that converts gestures into real-time text using advanced computer vision and deep learning techniques. By leveraging OpenCV and Mediapipe for hand detection, as well as MobileNetV2 for gesture classification, we have developed a model capable of accurately recognizing multiple sign language gestures. The training was conducted on an augmented dataset to enhance the model's robustness. Additionally, an interactive interface was implemented to allow real-time gesture recognition via a webcam. The results show a high accuracy rate, exceeding 90%, demonstrating the efficiency of our approach. This system represents a significant step towards accessibility and could be expanded to recognize a larger vocabulary and improve translation fluidity.

**Keywords :** Sign Language, Gesture Recognition, Computer Vision, Deep Learning, Automatic Translation.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Méthodologie</b>	<b>2</b>
2.1	Création d'une Base de Données Annotée . . . . .	2
2.2	Détection de la Région d'Intérêt (ROI) . . . . .	3
2.3	Extraction des Landmarks . . . . .	3
2.4	Prétraitement des Images . . . . .	4
2.5	Conception et Entraînement du Modèle . . . . .	4
2.6	Équilibrage du Dataset . . . . .	5
2.7	Stratégies d'Entraînement . . . . .	5
2.8	Test et Évaluation . . . . .	6
2.9	Résultats et Analyse . . . . .	6
2.10	Déploiement et Détection en Temps Réel . . . . .	7
<b>3</b>	<b>Conclusion</b>	<b>8</b>

# Table des figures

2.1	Exemple d'images des signes du dataset . . . . .	2
2.2	Exemple d'images après data augmentation . . . . .	2
2.3	Détection de la Région d'Intérêt (ROI) . . . . .	3
2.4	Exemple des images générées avec les landmarks . . . . .	4
2.5	Équilibrage du Dataset . . . . .	5
2.6	Distribution des Classes pour l'Entraînement et la Validation . . . . .	5
2.7	Arrêt de l'entraînement par le callback . . . . .	6
2.8	Évolution des performances du modèle au cours de l'entraînement . . . . .	6
2.9	Accuracy et Loss pour l'Entraînement et la Validation . . . . .	7
2.10	Détection en Temps Réel . . . . .	7

# Glossaire

**Apprentissage profond (Deep Learning)** : Sous-domaine de l'intelligence artificielle basé sur les réseaux de neurones artificiels permettant aux machines d'apprendre à partir de grandes quantités de données.

**Computer Vision** : Branche de l'intelligence artificielle qui permet aux ordinateurs d'interpréter et de comprendre les images et vidéos.

**Data Augmentation** : Technique utilisée pour enrichir un dataset en appliquant des transformations (rotation, zoom, etc.) afin d'améliorer la robustesse du modèle d'apprentissage.

**Langage des signes** : Langage visuel utilisant des mouvements des mains, des expressions faciales et la posture du corps pour communiquer, principalement utilisé par les personnes sourdes et malentendantes.

**Landmarks** : Points de repère détectés sur une image, en particulier sur une main ou un visage, permettant d'extraire des caractéristiques pour la reconnaissance de gestes ou d'émotions.

**Mediapipe** : Bibliothèque développée par Google permettant la détection et le suivi de la main, du visage et d'autres objets en temps réel à l'aide de modèles de vision artificielle.

**MobileNetV2** : Modèle de réseau de neurones optimisé pour les appareils à faible puissance, utilisé pour la classification d'images et la reconnaissance des objets.

**OpenCV (Open Source Computer Vision)** : Bibliothèque open-source de vision par ordinateur utilisée pour le traitement d'images et de vidéos.

**Reconnaissance des gestes** : Processus d'identification automatique des mouvements des mains et du corps pour comprendre et interpréter des actions spécifiques.

**ROI (Region of Interest)** : Partie spécifique d'une image extraite pour l'analyse, comme une main dans une vidéo pour la reconnaissance des signes.

**TensorFlow/Keras** : Frameworks populaires d'apprentissage profond permettant de construire, entraîner et déployer des modèles de réseaux de neurones.

**Transfert d'apprentissage** : Technique de machine learning utilisant un modèle pré-entraîné sur un grand dataset et l'adaptant à une tâche spécifique avec un entraînement supplémentaire.

# 1 Introduction

La communication est essentielle dans les interactions humaines, mais les personnes sourdes et malentendantes rencontrent souvent des difficultés pour se faire comprendre par ceux qui ne maîtrisent pas le langage des signes. Ce projet vise à développer un système capable de traduire les signes en texte en temps réel à partir d'une vidéo, facilitant ainsi l'accessibilité et l'inclusion sociale. Contrairement aux approches basées sur la reconnaissance des lettres de l'alphabet, notre solution identifie directement des mots complets afin d'améliorer la fluidité et la rapidité de la traduction.

Pour y parvenir, nous utilisons des techniques avancées de vision par ordinateur et d'apprentissage profond. OpenCV et Mediapipe permettent la détection et l'analyse des mouvements des mains, tandis que TensorFlow/Keras est employé pour classifier les signes. L'entraînement du modèle repose sur une base de données enrichie grâce à la data augmentation, et MobileNetV2 est utilisé pour garantir une reconnaissance efficace et rapide. Enfin, une interface interactive a été mise en place pour permettre une détection en temps réel.

## 2 Méthodologie

### 2.1 Crédation d'une Base de Données Annotée

Pour entraîner un modèle de reconnaissance des signes, une base de données d'images annotées a été constituée. Chaque classe correspond à un mot en langue des signes et est représentée par un dossier spécifique contenant plusieurs images de ce signe. Comme illustré dans l'image fournie, le dossier principal nommé test regroupe dix sous-dossiers, chacun correspondant à un mot particulier : Amour, Bien, Cool, Force, No, Parfait, Pause, Respect, Stop, Un.



FIGURE 2.1 – Exemple d'images des signes du dataset

Afin d'assurer une diversité suffisante et améliorer la robustesse du modèle, la technique de data augmentation a été appliquée sur les images existantes. Cette méthode consiste à générer des variations artificielles des images d'origine en appliquant des transformations telles que la rotation, la translation, le zoom et le retournement horizontal. Ces images augmentées ont été enregistrées dans un dossier séparé nommé augmented\_data, garantissant ainsi un dataset plus riche et mieux équilibré pour l'entraînement du modèle.[1]



FIGURE 2.2 – Exemple d'images après data augmentation

## 2.2 Détection de la Région d'Intérêt (ROI)

La détection des mains constitue une étape essentielle pour isoler la zone d'intérêt avant la reconnaissance des signes. Pour cela, OpenCV et Mediapipe ont été utilisés afin d'identifier automatiquement les mains dans une image ou une vidéo en temps réel. Grâce à Mediapipe Hands, un modèle de détection basé sur un réseau neuronal est appliqué pour repérer la position des mains et générer une boîte englobante autour de celles-ci.

Une fois la main détectée, la région d'intérêt (ROI) est extraite afin d'éliminer les éléments non pertinents de l'image. Cette technique permet d'améliorer la précision du modèle en réduisant les bruits d'arrière-plan et en focalisant l'analyse uniquement sur la main. La détection est effectuée avec un seuil de confiance ajustable, assurant ainsi une robustesse face aux variations d'éclairage et aux conditions environnementales.



FIGURE 2.3 – Détection de la Région d'Intérêt (ROI)

## 2.3 Extraction des Landmarks

Après l'extraction de la région d'intérêt, Mediapipe Hands est utilisé pour identifier les 21 landmarks clés de la main, représentant la position des articulations et l'orientation des doigts. Chaque point est défini par des coordonnées ( $x$ ,  $y$ ,  $z$ ), permettant d'analyser la structure et la posture de la main indépendamment de l'arrière-plan ou de la couleur de peau.[2]

Ces landmarks sont ensuite stockés sous forme de vecteurs et intégrés comme entrée du modèle d'apprentissage. Cette approche permet une reconnaissance plus précise des signes en se basant sur la géométrie et le mouvement des doigts plutôt que sur l'image brute. Ainsi, une meilleure généralisation du modèle est assurée, quelle que soit la condition de prise de vue.

Ci-dessous, un exemple des images générées avec les landmarks stockés dans le dossier landmarks.



FIGURE 2.4 – Exemple des images générées avec les landmarks

## 2.4 Prétraitement des Images

Avant d'être utilisées pour l'entraînement, les images sont redimensionnées à 256x256 pixels afin d'assurer une entrée homogène au modèle et d'optimiser les calculs. Une normalisation est ensuite appliquée en divisant chaque pixel par 255.0, ce qui permet de ramener les valeurs dans un intervalle entre 0 et 1, facilitant ainsi la convergence et améliorant la stabilité du modèle. Enfin, les images sont converties en tableaux de valeurs numériques, rendant leur traitement efficace par le réseau de neurones.

## 2.5 Conception et Entraînement du Modèle

Pour la reconnaissance des signes, un réseau de neurones convolutif basé sur **MobileNetV2** a été utilisé. Ce modèle est particulièrement adapté aux tâches de classification d'images en raison de sa légèreté et de son efficacité, même sur des appareils disposant de ressources limitées. Il repose sur un **transfert d'apprentissage**, où une version pré-entraînée de MobileNetV2 est utilisée et ajustée pour s'adapter aux spécificités de notre projet.

Le modèle est structuré de la manière suivante :

- Une **base pré-entraînée de MobileNetV2** avec ses premières couches gelées pour conserver les caractéristiques visuelles essentielles.
- Une **couche Flatten**, permettant de convertir les caractéristiques extraites en un vecteur exploitable.
- Des **couches denses avec activation ReLU**, favorisant l'apprentissage des relations complexes entre les features.
- Des **couches Dropout**, réduisant le risque de sur-apprentissage en désactivant aléatoirement certains neurones pendant l'entraînement.
- Une **couche de sortie avec activation softmax**, assurant la classification des signes en attribuant une probabilité à chaque classe.

L'optimisation du modèle est réalisée à l'aide de l'algorithme **Adam**, connu pour son efficacité dans la mise à jour des poids et la réduction de la fonction de coût. La fonction de perte choisie est **categorical\_crossentropy**, adaptée aux problèmes de classification

multi-classes. L’entraînement est effectué sur un jeu de données divisé en **80 % pour l’apprentissage et 20 % pour la validation**, avec un suivi des performances pour éviter le sur-apprentissage.

## 2.6 Équilibrage du Dataset

Un dataset déséquilibré peut entraîner une mauvaise généralisation du modèle, certaines classes étant mieux représentées que d’autres. Pour éviter ce problème, une **analyse du nombre d’images par classe** a été réalisée. Si une classe contenait significativement moins d’images que les autres, des **images supplémentaires ont été générées** grâce à la technique de **data augmentation** afin de rééquilibrer le jeu de données.

Les transformations appliquées incluent la **rotation**, qui permet de simuler différentes orientations des mains, ainsi que le **zoom**, afin d’ajuster l’échelle des signes sans en altérer le sens. De plus, la **translation** a été utilisée pour déplacer légèrement les images et ainsi mieux représenter les variations naturelles des gestes, tandis que le **retournement horizontal** a contribué à diversifier le dataset sans modifier la signification des signes. Une fois l’équilibrage effectué, une **distribution homogène des classes** a été obtenue, garantissant un entraînement plus efficace et une meilleure capacité du modèle à reconnaître tous les signes avec la même précision.

```
Nombre d'images par classe avant équilibrage : {'Amour': 283, 'Bien': 282, 'Cool': 282, 'Force': 235, 'Parfait': 298, 'Pause': 274, 'Respect': 287
, 'Stop': 293, 'Un': 277, 'no': 238}
Augmentation de la classe 'Amour' (283 → 298 images)
Augmentation de la classe 'Bien' (282 → 298 images)
Augmentation de la classe 'Cool' (282 → 298 images)
Augmentation de la classe 'Force' (235 → 298 images)
Augmentation de la classe 'Pause' (274 → 298 images)
Augmentation de la classe 'Respect' (287 → 298 images)
Augmentation de la classe 'Stop' (293 → 298 images)
Augmentation de la classe 'Un' (277 → 298 images)
Augmentation de la classe 'no' (238 → 298 images)

Équilibrage du dataset terminé !
```

FIGURE 2.5 – Équilibrage du Dataset

## 2.7 Stratégies d’Entraînement

L’entraînement du modèle a été réalisé sur un ensemble de données divisé en **80 % pour l’apprentissage et 20 % pour la validation**. Cette répartition permet d’assurer une phase d’apprentissage efficace tout en conservant un échantillon représentatif pour évaluer la capacité de généralisation du modèle. Pour optimiser l’apprentissage et éviter les problèmes de sur-apprentissage (*overfitting*), plusieurs stratégies ont été mises en place.

```
Distribution des classes dans le train : Counter({0: 239, 8: 239, 5: 239, 1: 239, 6: 238, 9: 238, 2: 238, 7: 238, 4: 238, 3: 238})
Distribution des classes dans le val : Counter({3: 60, 4: 60, 2: 60, 9: 60, 7: 60, 6: 60, 0: 59, 5: 59, 8: 59, 1: 59})
```

FIGURE 2.6 – Distribution des Classes pour l’Entraînement et la Validation

Tout d’abord, l’**Early Stopping** a été utilisé afin d’interrompre l’entraînement si la performance sur l’ensemble de validation se dégrade après un certain nombre d’époques

consécutives. Ensuite, la technique **ReduceLROnPlateau** a été appliquée, permettant de réduire le taux d'apprentissage si l'amélioration du modèle stagne, évitant ainsi des mises à jour inutiles des poids. Enfin, un **callback personnalisé** a été mis en place pour arrêter l'entraînement dès qu'un seuil prédéfini de précision ou de perte est atteint, garantissant ainsi un modèle optimisé sans nécessiter un nombre excessif d'époques d'entraînement.

```
Epoch 19/30
75/75 - 0s 526ms/step - accuracy: 0.9377 - loss: 0.1790
Arrêt de l'entraînement : Loss 0.1768 est en dessous de 0.2
75/75 - 48s 642ms/step - accuracy: 0.9377 - loss: 0.1790 - val_accuracy: 0.9047 - val_loss: 0.3225 - learning_rate: 1.0000e-05
Model training completed and saved.
```

FIGURE 2.7 – Arrêt de l'entraînement par le callback

## 2.8 Test et Évaluation

L'évaluation du modèle a été réalisée en observant l'évolution des métriques **accuracy** (précision) et **loss** (perte) tout au long de l'entraînement. Ces courbes permettent d'analyser le comportement du modèle et d'identifier d'éventuelles anomalies comme le sur-apprentissage ou un manque de convergence.

Les performances ont été validées sur un **ensemble de test indépendant**, où plusieurs métriques ont été utilisées pour mesurer la qualité des prédictions. L'**accuracy finale** obtenue sur l'ensemble de validation donne un indicateur global de la capacité du modèle à reconnaître correctement les signes. Enfin, l'analyse des **courbes de perte et de précision** a permis d'évaluer la stabilité du modèle et d'optimiser les paramètres pour améliorer la reconnaissance des signes.

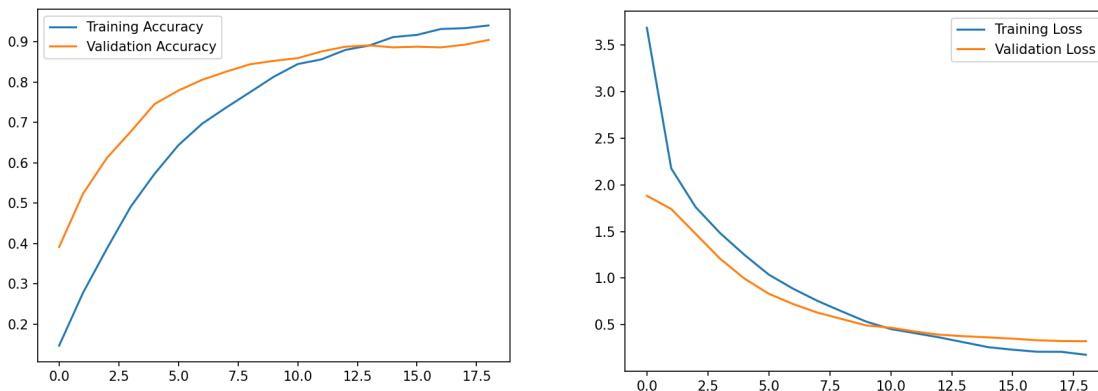


FIGURE 2.8 – Évolution des performances du modèle au cours de l'entraînement

## 2.9 Résultats et Analyse

Les résultats obtenus montrent que le modèle atteint une **précision élevée sur l'ensemble de validation**, confirmant sa capacité à reconnaître correctement les signes en langue des signes. L'entraînement du modèle s'est arrêté après **19 époques**, atteignant

une **accuracy de 93.77 %** et une **loss de 0.1790** sur l'ensemble d'entraînement, tandis que sur l'ensemble de validation, l'accuracy obtenue est de **90.47 %** avec une loss de **0.3225**.

```
Epoch 19/30
75/75 0s 526ms/step - accuracy: 0.9377 - loss: 0.1790
Arrêt de l'entraînement : Loss 0.1768 est en dessous de 0.2
75/75 48s 642ms/step - accuracy: 0.9377 - loss: 0.1790 - val_accuracy: 0.9047 - val_loss: 0.3225 - learning_rate: 1.0000e-05
Model training completed and saved.
```

FIGURE 2.9 – Accuracy et Loss pour l'Entraînement et la Validation

Toutefois, l'analyse des erreurs indique que certaines classes de signes très similaires peuvent être confondues, ce qui peut réduire la fiabilité dans des cas particuliers.

Pour améliorer la reconnaissance, plusieurs pistes peuvent être envisagées. L'ajout de **nouvelles images d'entraînement**, couvrant des angles et conditions d'éclairage variés, permettrait d'augmenter la robustesse du modèle. Par ailleurs, l'**utilisation de modèles de classification plus avancés**, tels que des réseaux de neurones basés sur les **transformers pour vision**, pourrait améliorer encore la précision et la capacité du modèle à distinguer les signes complexes.

## 2.10 Déploiement et Détection en Temps Réel

Une fois le modèle entraîné, il a été intégré dans un **script de détection en temps réel** utilisant **OpenCV** et **Mediapipe**. Le système fonctionne à partir d'une webcam, où chaque image capturée est traitée en temps réel pour détecter la main, extraire les landmarks et appliquer la classification.

L'interface de reconnaissance affiche la vidéo avec une **boîte englobant les mains**, accompagnée du mot correspondant au signe reconnu ainsi que de son **niveau de confiance**. L'utilisateur peut ainsi voir immédiatement la transcription du signe en texte. Ce déploiement en temps réel permet une utilisation interactive du modèle, ouvrant la voie à des applications pratiques pour faciliter la communication avec les personnes sourdes et malentendantes.[3]

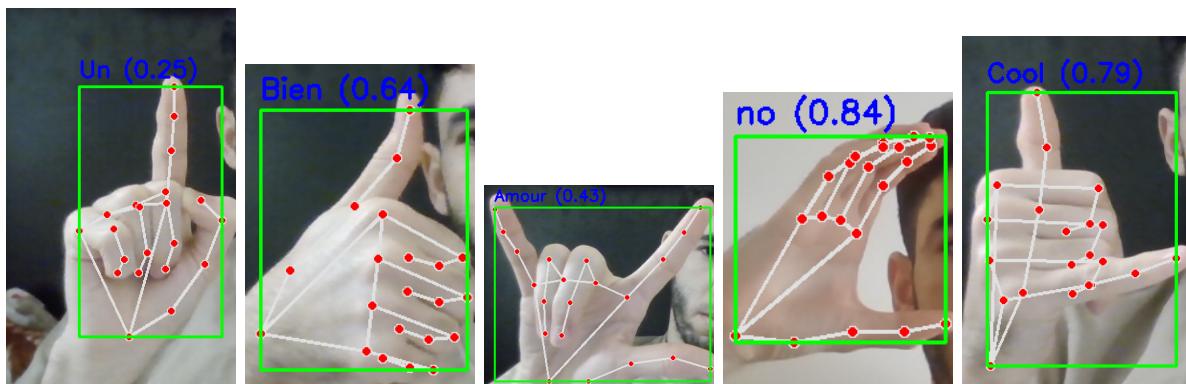


FIGURE 2.10 – Détection en Temps Réel

## 3 Conclusion

Ce projet a démontré la faisabilité d'un **traducteur automatique du langage des signes** basé sur l'intelligence artificielle et la vision par ordinateur. Grâce à des techniques avancées de **deep learning et de traitement d'images**, un modèle performant a été conçu pour reconnaître et traduire les signes en **temps réel**.

En automatisant la reconnaissance des gestes, ce système représente une avancée importante pour **l'accessibilité et l'inclusion** des personnes sourdes et malentendantes. Toutefois, des améliorations peuvent encore être apportées afin d'élargir les capacités du modèle, notamment pour mieux reconnaître les gestes en contexte dynamique et améliorer la précision sur un large éventail de signes. Ce projet ouvre ainsi la voie à des développements futurs dans le domaine de la traduction automatique du langage des signes.

# Bibliographie

- [1] <https://youtu.be/2fXJe9YqXgU?si=liy7mWZTeVE34TX1/>.
- [2] [https://www.youtube.com/watch?v=MJCSjXepaAM&t=480s&ab\\_channel=Computervisionengineer/](https://www.youtube.com/watch?v=MJCSjXepaAM&t=480s&ab_channel=Computervisionengineer/).
- [3] <https://github.com/computervisioneng/sign-language-detector-python/>.