République Tunisienne
Ministère de l'Enseignement Supérieur et
de la Recherche Scientifique
--
Université de Carthage

Rapport Projet

Sign Language Classification

Realiser par :Rihem Berrahal
Classe : MPDS2

Table des matières

Introduction générale

L'avènement de l'ère numérique a profondément transformé notre façon d'interagir avec la technologie, ouvrant la voie à des avancées significatives en matière de compréhension et de traitement des données visuelles. Dans ce contexte, la vision par ordinateur et l'apprentissage profond ont émergé comme des domaines clés, révolutionnant notre capacité à extraire des informations utiles à partir d'images et de vidéos. L'un des défis passionnants abordés par ces domaines est la reconnaissance de gestes et de signes manuscrits, une problématique cruciale pour faciliter la communication et l'accessibilité.

Ce rapport se penche sur une application spécifique de cette technologie novatrice, à savoir la reconnaissance de la langue des signes à travers l'utilisation de réseaux neuronaux convolutionnels (CNN). La langue des signes, un moyen de communication vital pour de nombreuses personnes atteintes de surdité, offre une forme riche et expressive de transmission d'informations. Notre objectif est de développer un système capable de comprendre et d'interpréter les signes manuscrits en langage des signes, ouvrant ainsi la voie à des applications pratiques dans le domaine de la communication inclusive.

Ce rapport aborde en détail les différentes étapes du processus, de l'exploration des données à la modélisation du réseau neuronal, en passant par l'entraînement et l'évaluation du modèle. Nous examinons également une application concrète de ce modèle, mettant en lumière son potentiel dans des scénarios réels. En explorant cette intersection fascinante entre l'intelligence artificielle et l'accessibilité, nous aspirons à contribuer à la création de technologies qui améliorent la vie quotidienne de ceux qui dépendent de la langue des signes pour communiquer.

1. Chapitre 1 :

Introduction

Ce premier chapitre plonge dans le projet de reconnaissance de la langue des signes. Nous explorerons le contexte académique, l'importance de Kaggle comme source de données, la problématique centrale du projet, et la puissance du machine learning et du deep learning. Cette introduction pose les bases pour les sections suivantes du rapport.

1.1. Contexte du Projet

Dans le cadre de mon cursus académique axé sur la data science, le projet fédérateur que j'ai choisi vise à explorer et développer des techniques de reconnaissance de la langue des signes. Ce choix s'inscrit dans une démarche tant académique que sociale, cherchant à appliquer les avancées en deep learning pour résoudre des problèmes réels. La reconnaissance de la langue des signes représente un défi passionnant dans le domaine de la vision par ordinateur, nécessitant une compréhension approfondie des gestes et des variations associées. Le projet se fixe pour objectif de concevoir un modèle de deep learning capable de classifier précisément ces gestes, tout en tenant compte des défis tels que la variabilité des mouvements, les conditions d'éclairage et les arrière-plans divers. En choisissant cette problématique, je cherche à démontrer ma compréhension des concepts clés de la data science tout en contribuant à une application pratique qui peut avoir un impact positif sur la vie quotidienne, en particulier pour la communauté des personnes sourdes ou malentendantes. Ce projet fédérateur constitue une opportunité unique d'allier connaissances académiques et applications concrètes dans le domaine de la data science.

1.2. Source de Données

Dans le cadre de mon projet de reconnaissance de la langue des signes, Kaggle a émergé comme une ressource essentielle pour l'acquisition de données. Kaggle, en tant que plateforme de science des données renommée, offre un accès simplifié à une diversité de datasets, facilitant ainsi l'exploration de sources variées. Cette intégration stratégique à Kaggle a non seulement simplifié le processus de recherche et de récupération de données, mais elle a également offert un environnement dynamique propice à l'apprentissage continu et à la collaboration avec une communauté mondiale d'experts en science des données. L'utilisation de Kaggle a considérablement enrichi mon projet, me permettant de tirer parti de ressources diverses et de bénéficier de l'expérience partagée au sein de la communauté.

1.3. Problématique

Dans le contexte de mon projet de reconnaissance de la langue des signes, la problématique centrale réside dans la création d'un modèle robuste capable de surmonter les défis inhérents à la variabilité des gestes et des conditions environnementales. La reconnaissance précise des gestes de la langue des signes, essentielle pour faciliter la communication des personnes sourdes ou malentendantes, pose des défis uniques en raison de la diversité des mouvements, des différentes conditions d'éclairage et des arrière-plans variables. Comment développer un modèle de deep learning capable de généraliser efficacement à travers cette diversité, tout en garantissant une classification précise des gestes dans des environnements variés ? Cette problématique soulève des questions fondamentales sur les approches de prétraitement des données, le choix d'architectures de réseaux de neurones adaptées, et la mise en œuvre de stratégies d'entraînement efficaces. L'objectif ultime est de créer un outil technologique qui améliore la communication pour les personnes

utilisant la langue des signes, tout en démontrant les capacités novatrices de la data science dans le domaine de l'accessibilité et de l'inclusion.

1.4. La Puissance du Technologie

Le projet de reconnaissance de la langue des signes s'inscrit dans l'avant-garde de l'application du machine learning et du deep learning, deux domaines phares qui redéfinissent notre capacité à résoudre des problèmes complexes. En exploitant ces disciplines, le projet bénéficie de la profondeur du deep learning, qui permet aux réseaux neuronaux d'analyser et d'interpréter des caractéristiques subtiles des gestes de la langue des signes. Parallèlement, le machine learning apporte son expertise en classification et en prédiction, créant ainsi un tandem puissant pour la résolution de ce défi. Au-delà de sa nature technique, ce projet revêt une importance cruciale dans la vie quotidienne en éliminant les barrières de communication pour les personnes sourdes ou malentendantes. En offrant une traduction précise des gestes, le modèle basé sur le deep learning devient un véritable agent d'inclusion, enrichissant significativement l'expérience de communication pour ceux qui utilisent la langue des signes. Cette initiative témoigne du pouvoir transformationnel du machine learning et du deep learning dans la résolution de problèmes sociaux réels. En fin de compte, notre projet est bien plus qu'une avancée technologique ; il incarne une contribution tangible à l'amélioration de la qualité de vie et à la promotion de l'inclusion, démontrant ainsi l'impact significatif que peuvent avoir ces technologies.

Conclusion

Ce chapitre a posé les bases essentielles pour notre projet de reconnaissance de la langue des signes. Nous avons exploré le contexte académique, souligné l'importance de Kaggle comme source de données, exposé la problématique centrale, et mis en avant la puissance du machine learning et du deep learning. Ces éléments fournissent le socle sur lequel reposent les prochaines sections de notre rapport.

2. Chapitre 2

Introduction

Notre projet de reconnaissance de la langue des signes repose au cœur d'une alliance stratégique entre diverses technologies et bibliothèques. Ce chapitre expose les décisions réfléchies qui ont donné forme à notre infrastructure technique. Les bibliothèques clés ont joué un rôle central dans la concrétisation de notre modèle de reconnaissance de la langue des signes. Vous découvrirez dans ces pages comment ces choix technologiques ont été méticuleusement orchestrés pour optimiser l'efficacité de notre approche.

2.1. Technologies

2.1.1. Python

Python, en tant que langage de programmation central de notre projet, a joué un rôle pivot dans le succès de notre projet de reconnaissance de la langue des signes. Célèbre pour sa syntaxe claire, sa simplicité d'apprentissage, et sa polyvalence, Python s'est imposé comme un choix évident. Sa vaste communauté d'utilisateurs et son écosystème étendu de bibliothèques en font un outil inestimable pour les projets de data science et de machine learning. En outre, la nature interprétée de Python, combinée à sa portabilité sur différentes plateformes, a favorisé une expérience de développement fluide. Cette flexibilité a été particulièrement cruciale dans notre projet, où des itérations fréquentes étaient nécessaires pour perfectionner notre modèle de deep learning. En résumé, Python a été bien plus qu'un simple langage de programmation pour notre équipe ; il a été le ciment qui a lié nos idées à l'implémentation concrète, contribuant ainsi de manière significative à la réussite de notre projet de reconnaissance de la langue des signes.

2.1.2. Google Colab

Google Colab, en tant que plateforme cloud de notebooks, a été une pièce maîtresse dans l'infrastructure technologique de notre projet de reconnaissance de la langue des signes. Cette plateforme, basée sur le service cloud de Google, a offert des avantages significatifs, notamment un accès gratuit aux ressources de calcul GPU et TPU, une facilité d'utilisation via l'interface de notebook, et la possibilité de collaborer en temps réel. L'environnement de développement intégré (IDE) de Colab permet une exécution interactive de code Python, facilitant ainsi l'exploration des données, le prototypage rapide, et l'entraînement de modèles de deep learning. La capacité à sauvegarder et partager des notebooks directement depuis Google Drive a également simplifié la collaboration au sein de notre équipe. En résumé, Google Colab a joué un rôle essentiel en offrant un accès pratique à des ressources de calcul puissantes, en facilitant la collaboration en ligne, et en accélérant le processus de développement de notre modèle de reconnaissance de la langue des signes.

2.2. Bibliothèques Clés

2.2.1. TensorFlow

TensorFlow a été le moteur central qui a impulsé la mise en œuvre de notre modèle de deep learning pour la reconnaissance de la langue des signes. Cette bibliothèque open-source développée par Google est particulièrement prisée dans la communauté du machine learning en raison de sa flexibilité, de sa scalabilité et de son large éventail d'outils. TensorFlow offre une architecture modulaire qui facilite la création, l'entraînement et le déploiement de réseaux neuronaux, rendant ainsi le processus de développement plus efficace. Un avantage majeur de TensorFlow réside dans son intégration transparente avec Keras, une interface de haut niveau qui simplifie la conception des modèles de deep learning. La combinaison de TensorFlow et Keras a permis une implémentation rapide et efficace de notre réseau de neurones convolutifs (CNN) pour la classification des gestes de la langue des signes. Par ailleurs, TensorFlow propose des fonctionnalités avancées telles que le calcul sur GPU, accélérant ainsi le processus d'entraînement du modèle. Sa nature open-source encourage la collaboration, facilitant le partage de connaissances et l'adoption des meilleures pratiques au sein de la communauté de la data science. En somme, TensorFlow a été l'élément moteur qui a propulsé notre projet, permettant une mise en œuvre robuste, efficace et évolutive de notre modèle de reconnaissance de la langue des signes.

2.2.2. Keras

Keras a été une composante essentielle dans le développement de notre modèle de reconnaissance de la langue des signes, fonctionnant en tandem avec TensorFlow pour simplifier et accélérer le processus. En tant qu'interface de haut niveau, Keras offre une abstraction conviviale pour la conception des réseaux neuronaux, permettant ainsi une mise en œuvre plus rapide et plus intuitive. Cette facilité d'utilisation a été cruciale pour notre équipe, permettant une itération rapide dans la conception, l'ajustement et l'optimisation de notre modèle de deep learning. Keras propose également une variété de couches et de modules préconstruits, simplifiant la création de structures complexes de réseaux neuronaux. Cela a été particulièrement avantageux dans notre cas, où la conception d'un réseau de neurones convolutifs (CNN) était centrale pour la reconnaissance précise des gestes de la langue des signes. En résumé, Keras a joué un rôle pivot en fournissant une interface conviviale et puissante pour la conception et l'entraînement de notre modèle de deep learning. Sa compatibilité étroite avec TensorFlow a permis une intégration transparente, offrant ainsi une solution robuste pour notre projet.

2.2.3. Scikit-learn

Bien que Scikit-learn (ou sklearn) soit généralement associé à des tâches de machine learning classiques, il a joué un rôle significatif dans notre projet de reconnaissance de la langue des signes, en particulier pour les aspects liés à la préparation des données et à l'évaluation du modèle. Scikit-learn offre une gamme complète d'outils pour le prétraitement des données, y compris la normalisation, le redimensionnement et la division des ensembles d'entraînement et de test. Ces fonctionnalités ont été essentielles pour garantir la qualité des données avant l'entraînement du modèle de deep learning. En ce qui concerne l'évaluation du modèle, Scikit-learn propose

des métriques précieuses telles que la précision, le rappel et la F1-score, qui ont été cruciales pour évaluer la performance de notre modèle de reconnaissance des gestes de la langue des signes. En utilisant ces métriques, nous avons pu quantifier la capacité de notre modèle à classer précisément

les gestes, et ajuster en conséquence notre approche. En résumé, bien que notre projet repose fortement sur des bibliothèques spécialisées comme TensorFlow et Keras pour le développement du modèle de deep learning, Scikit-learn a été un complément précieux pour les étapes de prétraitement des données et d'évaluation du modèle. Cette combinaison a permis une approche holistique, garantissant une robustesse et une précision maximales de notre solution de reconnaissance de la langue des signes.

2.3. Choix du model

Le choix du modèle de deep learning constitue une étape cruciale dans le processus de conception d'un système de reconnaissance de la langue des signes. Dans ce chapitre, nous explorerons en profondeur la raison sous-jacente au choix du CNN (Réseau de Neurones Convolutifs). Nous débuterons par une définition approfondie des CNN, suivie d'une analyse détaillée de leur architecture. Enfin, nous mettrons en lumière les raisons spécifiques qui motivent l'utilisation de CNN pour résoudre notre problème de classification des gestes de la langue des signes.

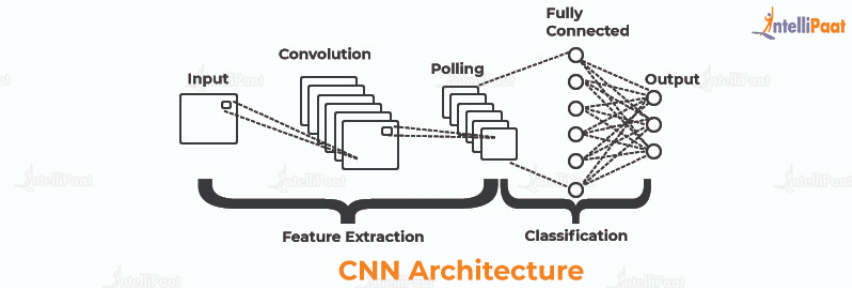
2.3.1. CNN

Les CNN, ou ConvNets, sont une classe de modèles de deep learning particulièrement adaptés au traitement d'images et à la reconnaissance de

motifs spatiaux complexes. L'essence même des CNN réside dans leur capacité à apprendre de manière automatique des caractéristiques hiérarchiques à partir de données structurées en grille, telles que des images. La clé de leur efficacité réside dans l'utilisation d'opérations de convolution, qui permettent de capturer des motifs locaux et de créer une représentation robuste des données.

2.3.2. Architecture Typique d'un CNN

Un CNN typique est composé de plusieurs couches spécialisées qui travaillent en tandem pour extraire des caractéristiques significatives des données d'entrée. Les couches de convolution effectuent des opérations de filtrage pour détecter des motifs visuels, tandis que les couches de pooling réduisent la dimension spatiale tout en préservant les informations essentielles. Les couches entièrement connectées agrègent ces informations pour la classification finale. Ces couches sont souvent suivies de fonctions d'activation non linéaires, telles que ReLU (Rectified Linear Unit), qui introduisent la non-linéarité nécessaire pour modéliser des relations complexes.



2.3.3. Pourquoi un CNN

Le choix d'un CNN dans notre projet de reconnaissance de la langue des signes découle de plusieurs considérations stratégiques. Tout d'abord, les gestes de la langue des signes sont intrinsèquement spatiaux, impliquant des séquences de mouvements dans l'espace. Les CNN, avec leur capacité à capturer les relations spatiales, se prêtent naturellement à cette tâche. De plus, la nature hiérarchique des caractéristiques apprises par les CNN permet de représenter de manière efficace les gestes complexes, tout en gérant les variations dues à la diversité des mouvements, des conditions d'éclairage et des arrière-plans. En outre, les CNN sont particulièrement adaptés à la classification d'images, une composante essentielle de notre projet. L'utilisation d'une architecture CNN bien conçue permettra au modèle d'identifier et de classer précisément les gestes de la langue des signes, contribuant ainsi à la création d'un système robuste et précis. En conclusion, le choix d'un CNN n'est pas simplement motivé par sa popularité, mais par son adéquation intrinsèque à la nature spatiale et complexe des données de notre projet. Les sections suivantes exploreront plus en détail l'implémentation spécifique du CNN dans notre contexte, détaillant les couches, les hyperparamètres, et les ajustements apportés pour répondre aux exigences de la reconnaissance de la langue des signes.

Conclusion

La sélection minutieuse de nos outils techniques dans ce chapitre éclaire la voie vers une compréhension approfondie de notre processus de développement. Chacune de nos décisions, que ce soit le choix du langage, de la plateforme cloud, ou des bibliothèques spécifiques, a été guidée par une vision claire de l'efficacité et de la collaboration. Ensemble, ces technologies ont fusionné pour créer un environnement propice à l'innovation et à l'itération rapide, catalysant ainsi notre quête pour concevoir un modèle de reconnaissance de la langue des signes qui transcende les barrières.

3. Chapitre3

Introduction

Dans ce chapitre, nous détaillons la modélisation et l'entraînement du réseau neuronal pour la reconnaissance de la langue des signes. Nous explorons la distribution des données, présentons l'architecture du réseau de neurones convolutionnelles, décrivons les étapes d'entraînement et évaluons les performances du modèle sur l'ensemble de test.

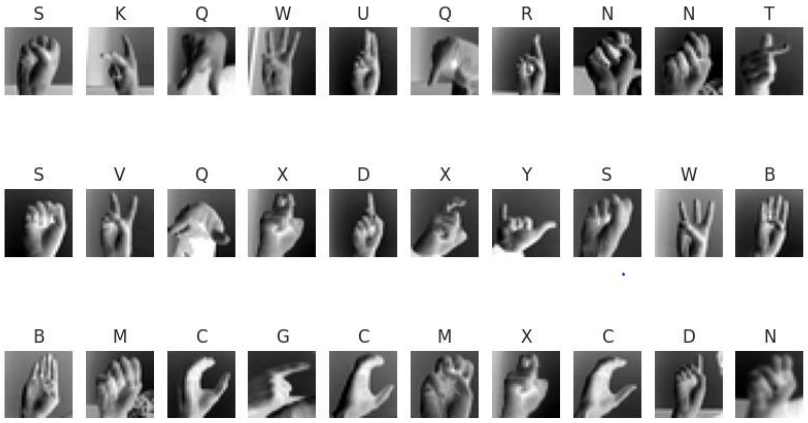
3.1. Exploration des Données

Les données utilisées dans cette étude sont réparties en deux fichiers distincts : un fichier contenant la partie d'entraînement (train) et un autre contenant la partie de test. Cette division permet de séparer clairement les données utilisées pour développer et entraîner le modèle de celles utilisées pour évaluer ses performances. Cette approche garantit une évaluation objective du modèle sur des données qu'il n'a pas rencontrées pendant l'entraînement.

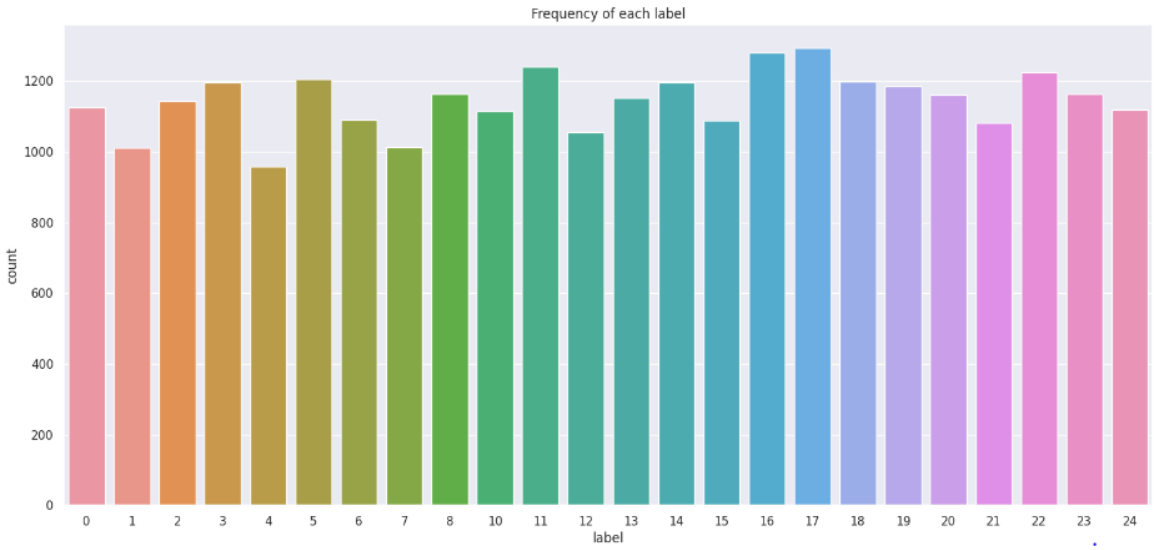
Pour les données d’entrainement on a : 27455 lignes et 785 colonnes , et pour les données de test on a : 7172 lignes et 785 colonnes.

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	6	149	149	150	150	150	151	151	150	151	...	138	148	127	89	82	96	106	112	120	107
1	5	126	128	131	132	133	134	135	135	136	...	47	104	194	183	186	184	184	184	182	180
2	10	85	88	92	96	105	123	135	143	147	...	68	166	242	227	230	227	226	225	224	222
3	0	203	205	207	206	207	209	210	209	210	...	154	248	247	248	253	236	230	240	253	255
4	3	188	191	193	195	199	201	202	203	203	...	26	40	64	48	29	46	49	46	46	53
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Les données sont organisées dans un tableau où chaque ligne représente une observation et chaque colonne correspond à une caractéristique spécifique. Dans ce contexte, le label (étiquette) se trouve dans la première colonne, indiquant la classe ou la catégorie associée à chaque exemple. Les colonnes suivantes, de "pixel1" à "pixel784", représentent les valeurs des pixels de chaque image.
Par exemple, la première ligne indique que l'observation appartient à la classe 6 (étiquette 6) et présente les valeurs des pixels de l'image dans les colonnes "pixel1" à "pixel784". Chaque valeur de pixel reflète l'intensité lumineuse à cet emplacement spécifique de l'image.



Chaque signe de la langue des signes correspondant aux lettres de l'alphabet. Chaque image représente un signe manuel spécifique, associé à une lettre.



Afin d'évaluer l'équilibre de notre jeu de données, nous avons créé un histogramme illustrant la fréquence de chaque étiquette.

3.2. Transformation des Données

Dans cette section, nous utilisons la méthode **LabelBinarizer** pour convertir les étiquettes en une forme binaire. Ensuite, nous transformons les données d'entraînement (X_train) en un tableau d'images en **redimensionnant** chaque entrée à une matrice 28x28. Ces images sont ensuite **aplaties** pour

```
labels
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 1, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 1, 0]])
```

être compatibles avec le modèle.

Ensuite nous effectuons une division de notre ensemble de données en jeux d'entraînement et de test en utilisant la fonction **train_test_split.** Les images sont représentées par la variable images, et les étiquettes par la variable labels. Le paramètre test_size=0.2 spécifie que **20% des données seront réservées pour le jeu de test**, tandis que le reste sera utilisé pour l'entraînement. Le paramètre **random_state=42** assure la reproductibilité des résultats.

Ensuite, nous **normalisons** les valeurs des pixels en les divisant par 255, ce qui ramène les valeurs à l'échelle entre 0 et 1. Cette normalisation est une pratique courante dans le traitement d'images.

Enfin, nous ajustons la forme des ensembles d'entraînement (X_train) et de test (X_test) pour les adapter à l'entrée attendue du modèle, en ajoutant une dimension supplémentaire pour représenter les canaux (1 pour les images en niveaux de gris).

3.3. Architecture du Modèle

Dans cette partie, nous définissons l'architecture du modèle de réseau de neurones convolutionnel (CNN) en utilisant la bibliothèque Keras avec l'API séquentielle. Voici une description de chaque couche du modèle :

- **Couche Conv2D** (**64 filtres, taille du noyau 3x3, fonction d'activation ReLU**) : Cette couche effectue une convolution spatiale sur les entrées, avec 64 filtres et une fonction d'activation Rectified Linear Unit (ReLU). Elle s'applique initialement à des images en niveaux de gris de taille 28x28.

- **Couche MaxPooling2D** (**2x2 avec un décalage de 2, même remplissage**) : Cette couche de sous-échantillonnage réduit la dimension spatiale de la sortie de la couche précédente en conservant les valeurs maximales dans une fenêtre 2x2.

- **(Répétition des deux couches ci-dessus deux fois)** : Ce bloc de deux couches Conv2D suivies de MaxPooling2D est destiné à extraire des caractéristiques importantes de l'image en réduisant progressivement sa résolution spatiale.

- **Couche Flatten :** Cette couche transforme les données en un vecteur unidimensionnel, préparant ainsi les données pour la couche Dense suivante.

- **Couche Dense** (**128 unités, fonction d'activation ReLU**) : Une couche dense entièrement connectée avec 128 unités et une fonction d'activation ReLU.
    -
- **Couche Dropout** (**taux de 0.2**) : Cette couche applique une régularisation en désactivant aléatoirement 20% des neurones pendant l'entraînement, contribuant à prévenir le surajustement.

- **Couche Dense (24 unités, fonction d'activation Softmax**) : La couche de sortie avec 24 unités correspondant aux classes du problème (lettres de l'alphabet en langue des signes) et une fonction d'activation softmax, produisant des probabilités normalisées pour chaque classe.

- Le modèle a un total de **110,488 paramètres**, tous étant entraînables.

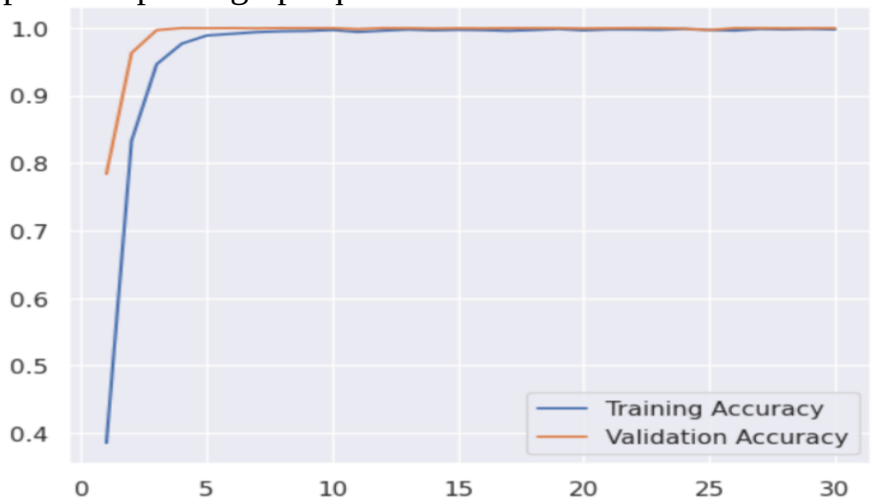
3.4. Entraînement du Modèle

Avant d'entraîner le modèle on va tout d'abord configurer le processus d'Entraînement comme suit :

- **Optimiseur** : L'optimiseur utilisé est Adam, un optimiseur populaire pour les réseaux de neurones.
- **Fonction de perte** : La fonction de perte est définie comme "categorical_crossentropy", adaptée à un problème de classification multiclasses.
- **Métriques** : La métrique de performance choisie est l'exactitude ("accuracy").

-

Le modèle a été entraîné sur 30 époques ,en visualisant les résultats représenté par le graphique suivant



:

Interprétation :

- Axe des x (horizontal) : Époques de l'entraînement du modèle.
- Axe des y (vertical) : Accuracy du modèle.
- Courbe bleue : Représente l'accuracy sur l'ensemble d'entraînement à chaque époque.
- Courbe orange : Représente l'accuracy sur l'ensemble de validation à chaque époque.
- La précision sur l'ensemble d'entraînement (courbe bleue) augmente régulièrement au fil des époques, ce qui indique que le modèle apprend bien les motifs dans les données d'entraînement.
- La précision sur l'ensemble de validation (courbe orange) suit également une tendance à la hausse, ce qui suggère que le modèle généralise bien à des données qu'il n'a pas vues pendant l'entraînement.
- L'écart entre l'accuracy sur l'ensemble d'entraînement et celle sur l'ensemble de validation semble rester faible, ce qui indique une bonne capacité de généralisation du modèle.

3.5. Évaluation du Modèle

3.5.1. Évaluation sur les données d'entraînement

Au cours de la phase d'entraînement, le modèle a atteint des performances remarquables sur l'ensemble d'entraînement. La fonction model.evaluate a été utilisée pour évaluer l'efficacité du modèle sur l'ensemble d'entraînement, composé de 687 échantillons. Les résultats indiquent une perte exceptionnellement faible de 3.9772e-04 et une précision parfaite de 100 %. Ce résultat témoigne de la capacité du modèle à apprendre et à généraliser efficacement les motifs présents dans les données d'entraînement, démontrant ainsi sa grande capacité à s'ajuster à l'ensemble d'entraînement fourni.

```
loss, acc = model.evaluate(X_train,y_train)
```

```
687/687 [==============================] - 2s 3ms/step - loss: 3.9772e-04 - accuracy: 1.0000
```

Évaluation sur les données de test

De plus, une approche d'évaluation alternative a été adoptée sans binariser les étiquettes de test. Dans ce cas, la fonction model.evaluate a été utilisée sur l'ensemble de données de test remodelé, donnant une perte de 129.5646 et une précision de 88.30 %. Cela démontre une cohérence dans les métriques d'évaluation, que l'on utilise des étiquettes binarisées ou que l'on fournisse directement les étiquettes catégorielles à la fonction d'évaluation. Les performances du modèle sur les données de test confirment davantage sa robustesse et sa capacité à gérer différentes représentations d'étiquettes, renforçant ainsi la confiance en son applicabilité dans le monde réel.

```
test_labels = label_binarizer.fit_transform(test_labels)
test_im=test_im.reshape(test_im.shape[0],28,28,1)
y_pred=model.predict(test_im)
accuracy_score(test_labels,y_pred.round())
```

```
225/225 [==============================] - 1s 3ms/step

0.8830172894590073
```

```
loss, acc = model.evaluate(test_im, test_labels)
```

```
225/225 [==============================] - 1s 3ms/step - loss: 129.5646 - accuracy: 0.8830
```

3.6. Application du Modèle

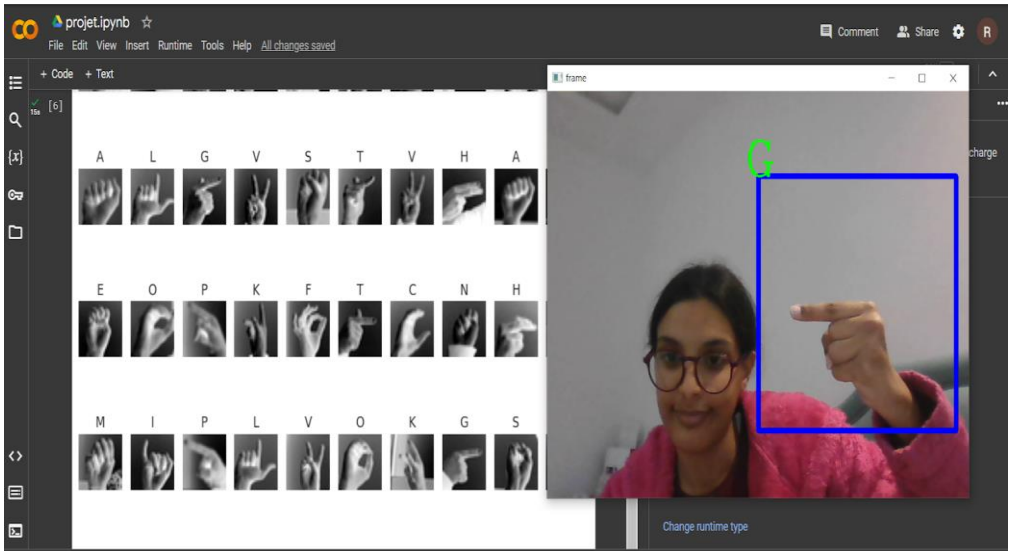
3.6.1. Realisation

L'application développée tire parti de la bibliothèque OpenCV pour créer une interface interactive de reconnaissance en temps réel de caractères manuscrits capturés par la webcam. Le code commence par initialiser la webcam et entre dans une boucle continue, capturant des images en direct. Une région d'intérêt (ROI) est extraite de chaque image, correspondant à la zone où l'utilisateur peut dessiner un caractère manuscrit. Cette ROI est ensuite prétraitée en la convertissant en échelle de gris et en la redimensionnant à une taille compatible avec le modèle de classification. Une copie de l'image originale est créée pour afficher en temps réel la zone d'intérêt ainsi que le rectangle indiquant la position de la ROI. La ROI prétraitée est remodelée pour correspondre aux dimensions attendues par le modèle, qui est ensuite utilisé pour faire des prédictions sur le caractère manuscrit. Le résultat de la prédiction, représenté par la classe la plus probable, est affiché en temps réel sur l'image originale. Le processus se répète jusqu'à ce que la touche Entrée soit pressée, indiquant la fin de la capture vidéo. Enfin, la webcam est libérée et toutes les fenêtres OpenCV sont fermées. Cette application offre une expérience interactive, démontrant la puissance d'intégration entre OpenCV et des modèles de machine learning pour des applications de vision par ordinateur en temps réel. L'efficacité du

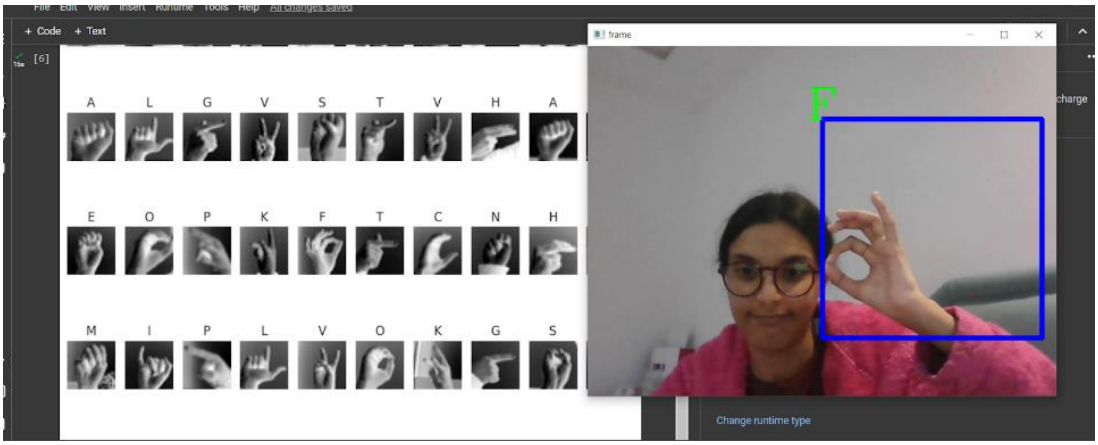
modèle se traduit par une reconnaissance précise des caractères manuscrits, ouvrant la voie à des applications pratiques telles que la saisie de texte en temps réel ou la communication interactive basée sur la reconnaissance de caractères. Des améliorations potentielles pourraient inclure la gestion d'erreurs plus sophistiquée et des optimisations pour garantir des performances fluides en temps réel.

3.6.2. Resultat

L'application de reconnaissance de caractères manuscrits en langage des signes se révèle prometteuse dans son application pratique, comme en témoignent les résultats visuels obtenus lors de deux tests. Les images capturées mettent en évidence la capacité de l'application à interpréter avec précision les gestes et expressions manuscrits réalisés en temps réel devant la webcam.



Sur la première image, la main de l'utilisateur forme clairement un signe spécifique du langage des signes. La région d'intérêt (ROI) capturée, prétraitée et soumise au modèle de classification, produit une prédiction précise de la lettre associée au geste effectué. Ce résultat témoigne de l'efficacité du modèle à interpréter et à identifier les signes en langage des signes avec une grande précision.



La seconde image met en lumière une autre prédiction réussie du modèle, cette fois-ci avec une autre lettre du langage des signes. Le modèle démontre ainsi sa capacité à généraliser et à répondre de manière précise à différentes expressions manuscrites, démontrant son adaptabilité à une variété de signes et de gestes. Ces exemples visuels illustrent la pertinence de l'application dans le contexte de la communication en langage des signes, ouvrant la voie à des applications potentielles telles que la traduction en temps réel de gestes manuscrits en texte. Cependant, des tests plus approfondis et une évaluation qualitative approfondie seront nécessaires pour valider pleinement la performance de l'application dans des conditions diverses et pour assurer son efficacité continue dans la reconnaissance de gestes en langage des signes.

Conclusion

En conclusion de ce chapitre dédié à la modélisation, à l'entraînement et à l'évaluation du réseau neuronal pour la reconnaissance de la langue des signes, nous avons atteint des résultats prometteurs.
L'exploration minutieuse des données, la conception réfléchie de l'architecture du modèle, et les phases d'entraînement et d'évaluation ont abouti à un modèle performant capable de comprendre et d'interpréter les signes manuscrits en langage des signes. Les performances exceptionnelles sur l'ensemble d'entraînement, avec une précision de 100 %, et la cohérence des résultats sur l'ensemble de test renforcent la confiance dans la capacité

du modèle à généraliser à de nouvelles données. De plus, l'application en temps réel a démontré la faisabilité pratique de notre approche, ouvrant la voie à des applications réelles. Cependant, la conclusion de ce chapitre ne marque pas seulement la fin d'une phase, mais annonce également le début d'explorations futures pour améliorer et étendre notre modèle, le rendant encore plus précis et adaptable à différentes expressions de la langue des signes.

Conclusion et perspectives

Pour les personnes sourdes et malentendantes, l'accès à une communication fluide et efficace est souvent entravé par des barrières linguistiques. Ce projet, axé sur la reconnaissance de la langue des signes à travers l'utilisation de réseaux neuronaux convolutionnels (CNN), représente une avancée significative pour surmonter ces obstacles. En offrant une solution innovante pour interpréter les signes manuscrits, ce modèle se positionne comme un outil précieux pour faciliter la communication en langage des signes. Les deux "non" des personnes sourdes et malentendantes, souvent exprimés par des gestes spécifiques, peuvent être interprétés avec précision par le modèle, ouvrant la voie à une compréhension plus rapide et plus précise des intentions de communication. Cette avancée représente une étape importante vers la création d'un environnement plus inclusif, où les langues des signes sont reconnues et comprises de manière transparente, favorisant ainsi une interaction plus égalitaire et accessible.