



# Rapport de projet

Implémentation d'un CNN pour  
prédire les panneaux routiers



**Fait par :**

**Bouarfa LAHMAR**

**Supervisé par :**

**M. K. ELMOUTAOUKIL**

# Sommaire

- 1. Introduction**
- 2. Objectifs**
- 3. Langage et bibliothèques utilisés**
- 4. Dataset utilisé**
- 5. Architecture du modèle CNN**
- 6. Résultats**
- 7. Conclusion**

---



# Introduction

La reconnaissance automatique des panneaux routiers joue un rôle crucial dans le domaine des systèmes d'aide à la conduite et des véhicules autonomes. En effet, identifier correctement ces panneaux permet d'améliorer la sécurité routière et d'assister le conducteur dans la prise de décisions. Avec l'essor de l'intelligence artificielle, les réseaux de neurones convolutionnels (CNN) se sont révélés particulièrement efficaces pour les tâches de classification d'images. Ce projet consiste à implémenter un CNN capable de classer les panneaux routiers à partir d'images. Afin de rendre cette technologie accessible et facile à utiliser, une interface graphique a également été développée pour permettre à l'utilisateur de charger une image et d'obtenir rapidement une prédiction du panneau présent. Ce rapport présente les différentes étapes de conception, d'implémentation et d'évaluation du modèle, ainsi que la mise en place de l'interface.



---

## 2. Objectifs

Ce projet s'inscrit dans le cadre du module Deep Learning et vise à mettre en pratique les connaissances théoriques acquises sur les réseaux de neurones convolutionnels (CNN). L'objectif principal est de concevoir, entraîner et évaluer un modèle CNN capable de reconnaître et classifier automatiquement les panneaux routiers à partir d'images, en exploitant les capacités d'extraction automatique de caractéristiques propres à ce type de réseau.

En complément, le projet ambitionne de développer une interface graphique conviviale qui facilite l'interaction avec le modèle, permettant à l'utilisateur de charger facilement des images, d'obtenir des prédictions en temps réel, et d'afficher les résultats de manière claire et compréhensible.

Ce travail pratique permet non seulement de renforcer la compréhension des concepts liés au deep learning et à la vision par ordinateur, mais aussi de développer des compétences en intégration de modèles d'apprentissage automatique dans des applications concrètes, un aspect essentiel pour leur déploiement dans des systèmes réels.

---

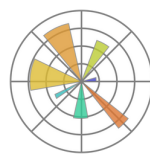
## 3. Langage et bibliothèques utilisés

Le projet a été développé en Python, un langage largement utilisé dans le domaine du deep learning et de la vision par ordinateur grâce à sa simplicité et à la richesse de son écosystème de bibliothèques.

Les principales bibliothèques utilisées sont :

- **NumPy** : pour la manipulation efficace des tableaux et des matrices de données.
- **Pandas** : pour la lecture, l'analyse et la gestion des données sous forme de tableaux.
- **Matplotlib** : pour la visualisation des données et des résultats, notamment les courbes d'apprentissage.
- **OpenCV (cv2)** : pour le traitement et la manipulation des images (lecture, redimensionnement, etc.).
- **scikit-learn** : pour la division du dataset en ensembles d'entraînement et de test (`train_test_split`) ainsi que pour l'évaluation des performances du modèle (précision, rappel, F1-score, matrice de confusion).
- **TensorFlow / Keras** : pour la création, l'entraînement et l'évaluation du réseau de neurones convolutionnel. Les modules utilisés incluent :
  - **Sequential** pour construire le modèle couche par couche,
  - **Conv2D, MaxPooling2D, Flatten, Dense, Dropout** pour définir l'architecture du CNN,
  - **to\_categorical** pour convertir les étiquettes en vecteurs de classes.

Ces outils permettent de développer l'ensemble du pipeline, depuis la préparation des données jusqu'à l'évaluation du modèle, de manière efficace et structurée.



---

## 4. Dataset utilisé

Pour ce projet, nous avons utilisé le GTSRB (German Traffic Sign Recognition Benchmark), un jeu de données largement utilisé pour les tâches de classification de panneaux de signalisation routière. Ce dataset contient plus de 50 000 images de panneaux répartis en 43 classes différentes, représentant divers types de signalisation (limitations de vitesse, interdictions, dangers, etc.).



Les images du dataset sont de tailles variables et en couleur. Un prétraitement a donc été nécessaire, comprenant :

- Le redimensionnement de toutes les images à une taille fixe (par exemple 32×32 pixels) pour les rendre compatibles avec l'entrée du modèle CNN.
- La normalisation des pixels (valeurs entre 0 et 1) afin d'accélérer la convergence lors de l'entraînement.
- La conversion des étiquettes en vecteurs one-hot avec `to_categorical` pour la classification multi-classes.

Le dataset a été divisé en deux parties principales :

- 80 % pour l'entraînement, utilisé pour ajuster les poids du réseau.
- 20 % pour les tests, utilisé pour évaluer la performance du modèle sur des données jamais vues.

Ce dataset est bien adapté à la tâche, car il est riche, diversifié et bien étiqueté, ce qui permet d'entraîner un modèle robuste et généralisable.

 Test  
 Train

05/22/2025 12:04 PM

File folder

05/22/2025 12:10 PM

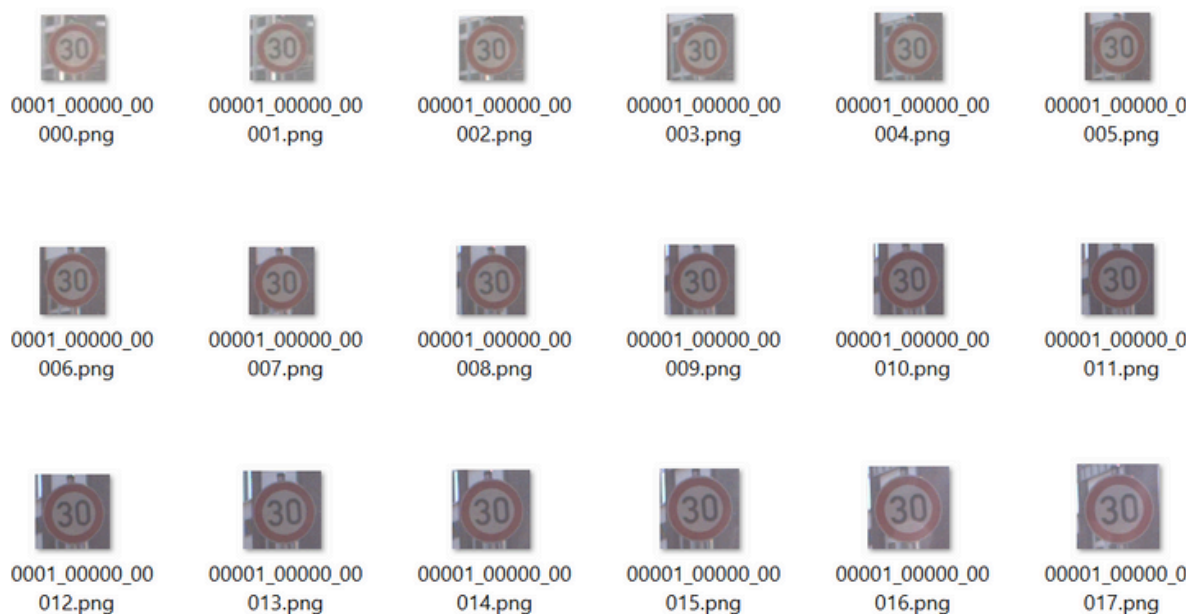
File folder

# Tableau des classes du dataset

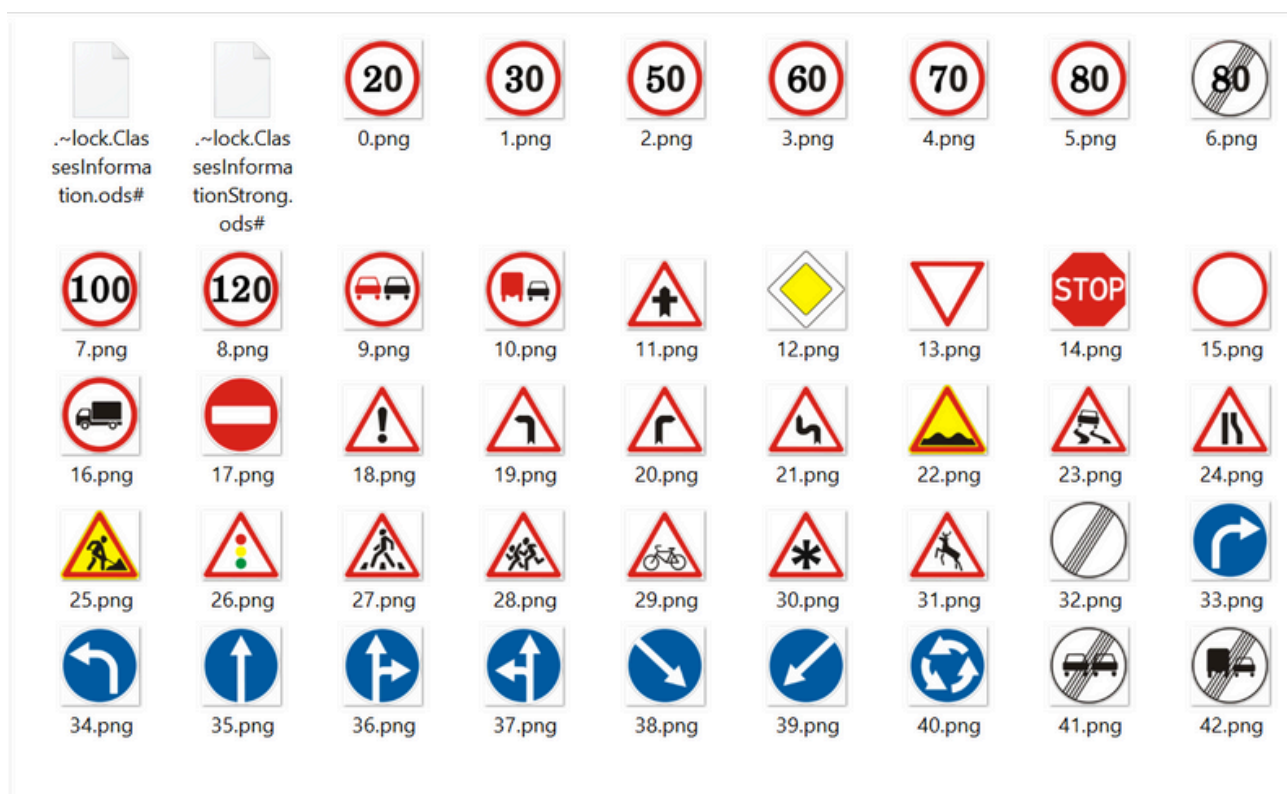
ClassId	Name
0	Speed limit (20km/h)
1	Speed limit (30km/h)
2	Speed limit (50km/h)
3	Speed limit (60km/h)
4	Speed limit (70km/h)
5	Speed limit (80km/h)
6	End of speed limit (80km/h)
7	Speed limit (100km/h)
8	Speed limit (120km/h)
9	No passing
10	No passing for vehicles over 3.5 metric tons
11	Right-of-way at the next intersection
12	Priority road
13	Yield
14	Stop
15	No vehicles
16	Vehicles over 3.5 metric tons prohibited
17	No entry
18	General caution
19	Dangerous curve to the left
20	Dangerous curve to the right
21	Double curve
22	Bumpy road
23	Slippery road
24	Road narrows on the right
25	Road work
26	Traffic signals
27	Pedestrians
28	Children crossing
29	Bicycles crossing
30	Beware of ice/snow
31	Wild animals crossing
32	End of all speed and passing limits
33	Turn right ahead
34	Turn left ahead
35	Ahead only
36	Go straight or right
37	Go straight or left
38	Keep right
39	Keep left
40	Roundabout mandatory
41	End of no passing
42	End of no passing by vehicles over 3.5 metri



Chaque classe du dataset contient de nombreuses images pour l'entraînement :



Voici tous les types de signaux utilisés pour entraîner le modèle :





---

## 5. Architecture du modèle CNN

Pour la classification des panneaux de signalisation, un réseau de neurones convolutionnel (CNN) a été conçu à l'aide de l'API Sequential de Keras. Ce type de réseau est particulièrement adapté aux données visuelles, car il permet de détecter automatiquement les caractéristiques importantes des images (bords, formes, textures, etc.).

L'architecture du modèle utilisée est la suivante :

1. Couche de convolution 1 :
  - 32 filtres de taille 3×3
  - Fonction d'activation : ReLU
  - Dimension d'entrée : (32, 32, 3) pour des images RGB redimensionnées
2. Couche de max pooling 1 :
  - Taille de filtre : 2×2
  - Permet de réduire la taille de l'image et d'extraire les caractéristiques dominantes
3. Couche de convolution 2 :
  - 64 filtres de taille 3×3
  - Fonction d'activation : ReLU
4. Couche de max pooling 2 :
  - Taille de filtre : 2×2
5. Flatten :
  - Transformation de la matrice 2D en vecteur 1D pour l'entrée des couches denses

---

6. Couche Dense (complètement connectée) :

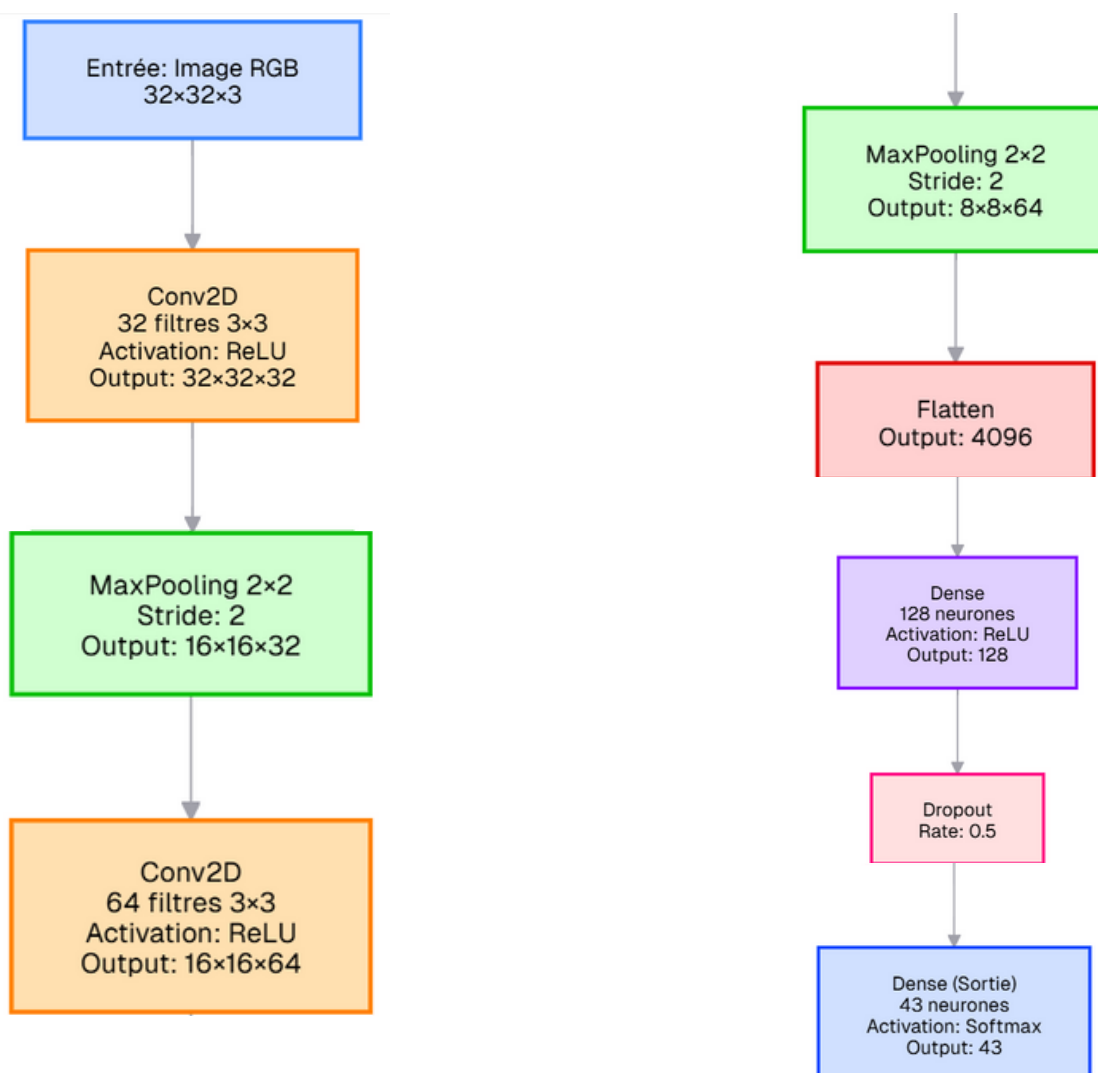
- 128 neurones
- Fonction d'activation : ReLU

7. Dropout :

- Taux de dropout : 0.5
- Permet de réduire le surapprentissage en désactivant aléatoirement certains neurones pendant l'entraînement

8. Couche de sortie :

- Nombre de neurones égal au nombre de classes (43)
- Fonction d'activation : softmax, adaptée à la classification multi-classes



---

## 6. Résultats du modèle CNN:

Après l'entraînement du modèle de réseau de neurones convolutif (CNN) sur l'ensemble de données d'images, plusieurs métriques ont été utilisées pour évaluer ses performances sur l'ensemble de test. Les résultats obtenus sont les suivants :

- **Exactitude (Accuracy)**

Le modèle a atteint une accuracy de 99,02 % sur l'ensemble de test. Cela signifie que le modèle a correctement prédit la classe de 99,02 % des images testées. Cette valeur élevée indique une très bonne capacité du modèle à généraliser les connaissances apprises lors de l'entraînement.

- **Rappel (Recall Macro)**

Le recall macro est de 98,61 %, ce qui démontre que, globalement, le modèle est capable de détecter correctement la majorité des exemples pertinents pour chaque classe. Le rappel est une métrique importante, surtout dans des contextes où il est crucial de minimiser les faux négatifs.

- **F1-Score Macro**

Le F1-score macro atteint 98,82 %, ce qui traduit un bon équilibre entre le rappel et la précision. Cela confirme que le modèle est performant à la fois dans l'identification des bonnes classes et dans la réduction des mauvaises classifications.

- **Analyse globale**

Les résultats obtenus démontrent que le modèle CNN est très performant et bien adapté à la tâche de classification d'images dans ce contexte. Le modèle parvient à distinguer efficacement les différentes classes du jeu de données avec une forte cohérence entre les métriques d'évaluation.

218/218 ————— 2s 9ms/step

✓ Accuracy : 0.9902

☑ Recall (sensibilité) par classe : [1. 0.97932817 0.99297424 0.95522388 0.99399399 0.9852071

1. 0.98867925 1. 0.99283154 1. 0.99082569

1. 1. 0.99315068 1. 1. 1.

1. 0.91666667 0.94736842 0.95 0.98305085 0.98684211

0.96610169 0.99622642 0.97169811 0.9047619 0.99 0.95

1. 0.98780488 1. 1. 1. 1.

0.98412698 1. 0.99463807 1. 1. 1.

1. ]

📊 Recall macro : 0.9861

🔗 F1-score par classe : [0.98113208 0.98955614 0.98834499 0.97338403 0.99548872 0.96661829

1. 0.99242424 0.99545455 0.99640288 1. 0.99539171

0.99853157 0.99751244 0.99656357 1. 1. 0.99731903

0.98623853 0.95652174 0.97297297 0.97435897 0.99145299 0.97402597

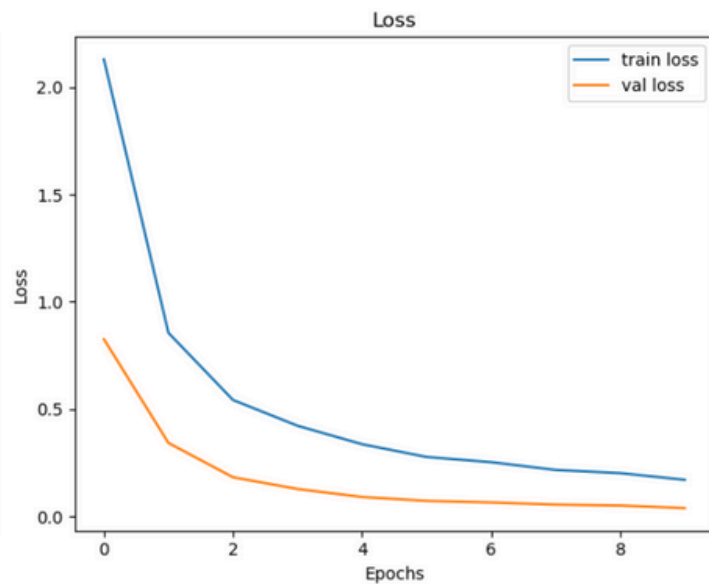
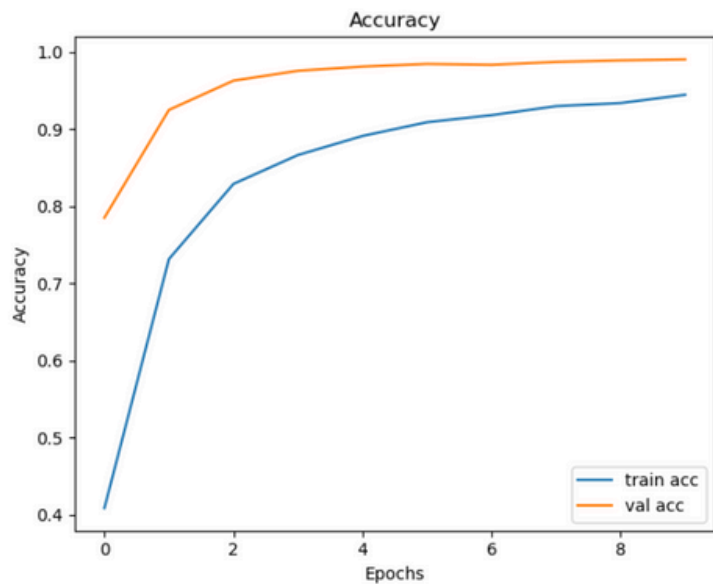
0.96610169 0.98876404 0.98564593 0.95 0.98019802 0.97435897

0.9689441 0.98181818 1. 0.9958159 0.99337748 0.99784946

0.992 1. 0.99597315 1. 1. 1.

1. ]

🚩 F1-score macro : 0.9882



---

## 7. Interface graphique

À la fin de l'entraînement et de l'évaluation du modèle CNN, j'ai sauvegardé le modèle entraîné au format .h5 à l'aide de la fonction `model.save('nom_du_modele.h5')`. Cette étape permet de réutiliser le modèle sans avoir à le réentraîner à chaque exécution, ce qui est essentiel pour une utilisation pratique.

Afin de faciliter l'utilisation du modèle par des utilisateurs non techniques, j'ai développé une interface graphique simple à l'aide de la bibliothèque Tkinter, intégrée à Python. Cette interface permet de tester facilement le modèle en important une image et en obtenant instantanément la prédiction de la classe correspondante.

--> Objectifs de l'interface :


- Offrir un accès simple et interactif au modèle.
- Permettre à l'utilisateur de charger une image de signal depuis son ordinateur.
- Afficher directement la classe prédite par le modèle.
- Afficher l'image sélectionnée dans l'interface pour plus de clarté.

→ Fonctionnalités de l'interface :

- Bouton de sélection d'une image locale (.jpg, .png, etc.).
- Prétraitement automatique de l'image sélectionnée pour respecter les exigences du modèle (redimensionnement à 32x32 pixels, normalisation, etc.).
- Chargement du modèle .h5 en mémoire et utilisation pour prédire la classe.
- Affichage du résultat dans une étiquette ou une zone de texte dédiée.
- Affichage graphique de l'image choisie dans l'interface.

Cette interface rend le modèle accessible de manière simple et intuitive, démontrant ainsi une application concrète et pratique de l'apprentissage automatique dans un environnement utilisateur.

## On charge l'image pour le test





École d'Ingénierie Digitale  
et d'Intelligence Artificielle

### Traffic Sign Classifier

Computer Vision (CNN) Project

Made by Bouarfa Lahmar





Click here to select an image

Predict Traffic Sign

#### Prediction Results


Select an image and click 'Predict'

Confidence: --

Top 3 Predictions:

1.	--	--
2.	--	--

On appuie sur le bouton Predict et les résultats s'affichent:





École d'Ingénierie Digitale  
et d'Intelligence Artificielle

### Traffic Sign Classifier

Computer Vision (CNN) Project

Made by Bouarfa Lahmar





Predict Traffic Sign

#### Prediction Results

Prediction Result:

**Speed limit (70km/h)**

Confidence: **95.99%**

Top 3 Predictions:

1.	Speed limit (70km/h)	95.99%
2.	Speed limit (20km/h)	2.09%

---

## 8. Conclusion

Ce projet avait pour objectif de concevoir, entraîner et évaluer un modèle de réseau de neurones convolutif (CNN) pour la classification d'images appartenant à différentes classes de signaux. À travers les étapes de prétraitement, de construction du modèle, d'entraînement et d'évaluation, nous avons pu démontrer l'efficacité de l'approche choisie.

Le modèle a atteint une précision de 99,02 %, accompagnée de très bonnes performances en termes de rappel et de F1-score, ce qui prouve sa robustesse et sa capacité à généraliser sur des données inédites. La simplicité de l'architecture utilisée, combinée à un bon équilibre des classes et à un prétraitement rigoureux, a permis d'obtenir des résultats remarquables sans recours à des modèles trop complexes.

→ Perspectives d'amélioration :

Pour aller plus loin, plusieurs pistes d'amélioration peuvent être envisagées :

- Augmenter la taille du dataset ou utiliser des techniques d'augmentation de données,
- Optimiser les hyperparamètres du modèle,
- Tester des architectures CNN plus avancées (comme VGG, ResNet, etc.),
- Appliquer le transfer learning avec des modèles pré-entraînés.

En conclusion, ce projet constitue une base solide pour le développement d'applications de reconnaissance visuelle dans des contextes réels, et ouvre la voie à des expérimentations plus poussées dans le domaine de la vision par ordinateur.