Rapport de Projet d'Apprentissage Profond : Détection et Reconnaissance des Panneaux de Signalisation Routière

SARHANE ABDELMOUHAIMEN
OURKIA ABDELHAKIM
BILAL AZDAD
EL HEYBA EL HEYBA

Département Sciences du Numérique - Deuxième année
2023-2024

Contents

1 Introduction

Dans ce projet, nous avons développé un modèle de classification des panneaux de signalisation en utilisant des techniques de deep learning. Notre objectif était de créer un modèle performant capable de détecter et de reconnaître différents types de panneaux de signalisation routière, tels que les panneaux de passage piéton, les panneaux d'arrêt "Stop", les panneaux de limitation de vitesse, les feux tricolores et les panneaux d'interdit.
Pour atteindre cet objectif, nous avons suivi un processus rigoureux de préparation des données, de conception et d'entraînement de modèles simples et avancés, et d'analyse détaillée des résultats obtenus. Ce rapport présente les étapes clés de notre travail, les techniques utilisées, les résultats obtenus et les perspectives d'amélioration.

Plus précisément, nous avons ciblé les cinq classes suivantes :

- Passage piéton
- Entrée interdite
- Limitation de vitesse
- Arrêt (“Stop”)
- Feu de circulation

2 La constitution de la base de données

2.1 Base de Données

Vous pouvez trouver notre base de données sur GitHub à l'adresse suivante : https://github.com/Abdelmouha
TrafficSignClassification/tree/main/dataGitHub - Base de Données.

Méthodologie

Pour obtenir un ensemble de données d'images de panneaux de signalisation routière, nous avons utilisé une approche en deux étapes :

2.1.1 Collecte des Images via Web Scraping

Nous avons utilisé Selenium et BeautifulSoup pour automatiser la collecte d'images à partir de sources en ligne telles que Google Images. Notre script de web scraping est disponible sur GitHub à l'adresse suivante : https://github.com/Abdelmouhaimen/DL-TrafficSignClassification/blob/main/

2.1.2 Traitement des Images

Après avoir collecté les images, nous avons effectué plusieurs étapes de prétraitement :

- **Suppression des Images de Petite Taille :** Nous avons filtré les images dont la hauteur ou la largeur était inférieure à 50 pixels, car elles étaient susceptibles de ne pas être suffisamment informatives pour l'apprentissage.

- **Nettoyage Manuel :** Nous avons examiné manuellement les images restantes pour éliminer celles qui ne correspondaient pas à nos attentes ou qui étaient de mauvaise qualité. Cela a permis d'éliminer les images inutiles ou ambiguës de notre ensemble de données.

2.2 Méthodologie de Partitionnement

Pour structurer efficacement notre projet d'apprentissage profond, nous avons opté pour une approche de partitionnement des données en trois ensembles principaux : l'entraînement, la validation et le test. Cette démarche vise à optimiser l'apprentissage du modèle et à garantir une évaluation fiable de ses performances.

2.2.1 Ensemble d'Entraînement (70% des données)

Pourcentage choisi : 70%.
Cette part jeu de données est dédiée à l'apprentissage du modèle, notre modèle dispose d'une base de données conséquente pour apprendre à discriminer les différentes catégories de panneaux de signalisation. Ce pourcentage élevé assure une variété suffisante d'exemples, cruciale pour une généralisation robuste du modèle.

2.2.2 Ensemble de Validation (15% des données)

Pourcentage choisi : 15%.
L'ensemble de validation est utilisé pour ajuster les hyperparamètres du modèle et prévenir le surajustement. Il permet d'évaluer la performance du modèle sur des données non vues pendant l'entraînement, sans compromettre sa capacité à généraliser.

2.2.3 Ensemble de Test (15% des données)

Pourcentage choisi : 15%.
Cet ensemble sert à évaluer la performance finale du modèle. Ce pourcentage assure une estimation fiable de la performance du modèle tout en conservant une partie significative des données pour l'entraînement et la validation.

2.3 Pronostic sur le Problème et les Résultats

La classification des panneaux de signalisation routière représente un défi complexe. Les premiers résultats sont prometteurs, mais plusieurs obstacles doivent être surmontés pour obtenir une précision optimale.

2.3.1 Complexité du Problème

La variété des panneaux, les conditions variables de prise de vue et la possible superposition des panneaux dans les images rendent la tâche complexe.

2.3.2 Pronostic sur les Résultats

Nous anticipons une précision modérée à élevée, avec une estimation minimale de 80%. La performance dépendra de la qualité des données, du choix de l'architecture du modèle et de l'utilisation de techniques d'apprentissage appropriées.

2.4 Script de Chargement des Données

Vous pouvez trouvez le script Python pour charger nos données sur GitHub à l'adresse suivante : https://github.com/Abdelmouhaimen/DL-TrafficSignClassification/blob/main/chargement$_d$onnees.ipynbC

2.5 Images de notre Base de Données

Figure 1: Panneau de passage piéton

Figure 2: Panneau d'interdiction

Figure 3: Panneau de limitation de vitesse

Figure 4: Panneaux 1

Figure 5: Panneau d'arrêt "Stop"

Figure 6: Feu tricolore

Figure 7: Panneaux 2

3 Étapes de Résolution du Problème

Dans cette partie, nous décrirons en détail les différentes étapes que nous avons suivies pour résoudre notre problème de classification des panneaux de signalisation en utilisant des réseaux de neurones convolutionnels (CNN). Nous commencerons par la construction d'un modèle CNN de base, puis nous expliquerons l'utilisation de techniques d'augmentation de données pour améliorer la variabilité des images d'entraînement. Enfin, nous aborderons l'approche du fine-tuning en utilisant un réseau pré-entraîné pour obtenir de meilleures performances.

3.1 Pré-traitement

Description des Données: Les données utilisées sont des images en couleur de panneaux de signalisation. Initialement, nous avons choisi de travailler avec des images de taille 64x64 pixels, mais les résultats obtenus étaient moyens. Pour améliorer les performances, nous avons décidé de redimensionner les images à 128x128 pixels et de les stocker sous forme de tableaux NumPy. Cette approche a permis d'obtenir de meilleurs résultats. Les cinq classes cibles sont les suivantes : passage piéton, entrée interdite, limitation de vitesse, arrêt et feu de circulation.

Normalisation des données : Chaque pixel de l'image est normalisé en divisant par 255 car La normalisation des images avant leur traitement par un réseau de neurones convolutifs (CNN) est cruciale pour plusieurs raisons. En divisant chaque pixel par 255, les valeurs sont mises dans une plage de 0 à 1, ce qui améliore la stabilité numérique et prévient les problèmes de gradients explosifs ou évanescents. Cela est particulièrement important lorsque l'on utilise la descente de gradient, car cette méthode d'optimisation converge plus rapidement avec des entrées normalisées. De plus, la normalisation rend les données d'entrée plus uniformes, facilitant l'extraction de caractéristiques pertinentes par les couches convolutionnelles. Ainsi, cette étape améliore la performance globale du modèle en rendant l'apprentissage plus efficace et les résultats plus fiables.

3.2 Construction du Modèle CNN

3.2.1 Version 1 :

Pour commencer, nous avons construit un modèle CNN (Convolutional Neural Network) pour effectuer la classification des panneaux de signalisation. Ce modèle est composé de couches de convolution, de couches de pooling et de couches entièrement connectées. Les couches de convolution sont responsables de l'extraction des features des images, tandis que les couches de pooling réduisent la dimensionnalité des features extraites. Enfin, les couches entièrement connectées sont utilisées pour la classification des images.

Architecture détaillée du réseau :

Le modèle de réseau de neurones convolutif (CNN) utilisé pour la classification des panneaux de signalisation est construit avec la bibliothèque Keras. Il commence par une couche d'entrée acceptant des images de taille 128×128 pixels avec trois canaux de couleur (RGB). Ensuite, il comporte quatre couches convolutionnelles successives avec des filtres de tailles croissantes (32, 64, 96 et 128), chacune utilisant des noyaux de taille 3×3 et la fonction d'activation ReLU. La fonction ReLU est utilisée à la place de la fonction sigmoïde pour

éviter le problème de l'évanescence des gradients, car la dérivée de la fonction sigmoïde est limitée à $[0, \frac{1}{4}]$, ce qui diminue l'amplitude des gradients propagés à travers les couches du réseau de neurones. Après chaque couche convolutionnelle, une couche de max-pooling de taille 2×2 réduit la dimensionnalité spatiale, et une couche de dropout (à 25%) aide à prévenir le surapprentissage en désactivant aléatoirement un certain nombre de poids au cours du processus d'apprentissage, évitant ainsi que certains poids ne deviennent trop importants par rapport à d'autres. Après les couches convolutionnelles, le modèle intègre une couche de flattening pour transformer les matrices en vecteurs, suivie d'une couche dense avec 512 neurones et activation ReLU, et une autre couche de dropout (à 50%). Enfin, la couche de sortie utilise la fonction d'activation softmax pour classer les images en cinq catégories. Le modèle est compilé avec l'optimiseur Adam, qui adapte le momentum pour éviter les problèmes de stabilisation dans un minimum local, et utilise la perte d'entropie croisée catégorielle sparse et la métrique de précision catégorielle sparse.

Résumé de l'architecture :

- Couches convolutionnelles : 4 couches avec des filtres de taille 3×3
- Couches de pooling : MaxPooling après chaque couche de convolution
- Couche entièrement connectée : 1 couche avec 512 neurones
- Taux de dropout : 0.25 après chaque couche convolutionnelle et 0.5 après la couche entièrement connectée

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
dropout (Dropout)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
dropout_1 (Dropout)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 96)	55392
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 96)	0
dropout_2 (Dropout)	(None, 14, 14, 96)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	110720
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_3 (Dropout)	(None, 6, 6, 128)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 512)	2359808
dropout_4 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 5)	2565

Figure 8: nombre de parametres du model

Liste des hyperparamètres :

- Taille des filtres : 3×3
- Nombre de filtres : 32, 64, 96, 128
- Taux de dropout : 0.25, 0.5
- Taux d'apprentissage : Default
- Taille du batch : Default
- Nombre d'époques : 30 - 50

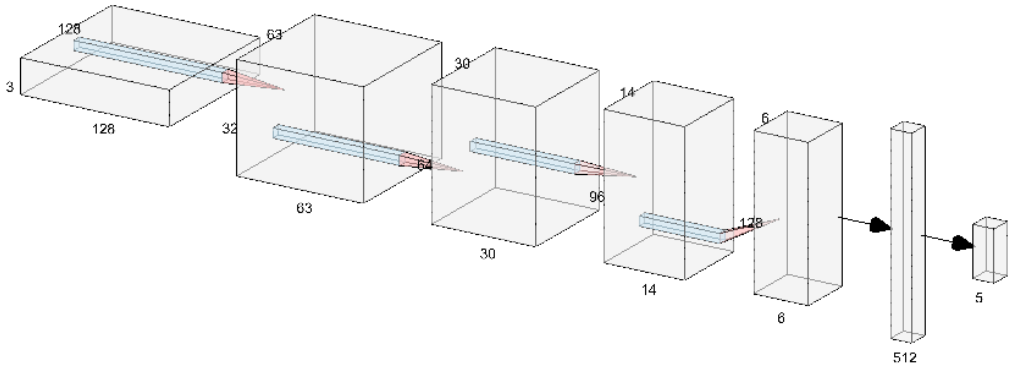


Figure 9: L'architecture CNN pour la classification d'images comprend des couches convolutives, des couches de pooling maximum et des couches entièrement connectées.

3.2.2 Verison 2

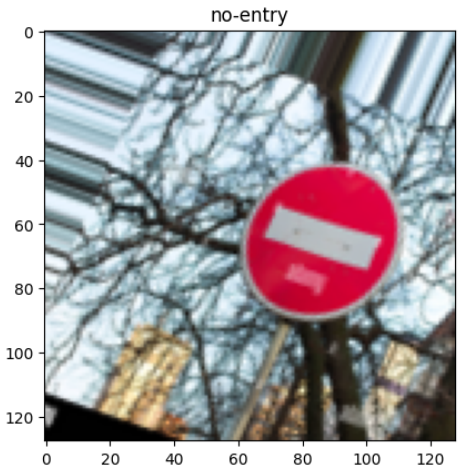
Architecture détaillée du réseau :

Dans cette version, nous avons utilisé la même architecture déjà expliquée dans la partie précédente. La seule différence est que nous avons introduit quelques approches de régularisation afin de limiter le surapprentissage et d'obtenir de meilleurs résultats. Ces approches comprennent l'utilisation de l'augmentation de données et des callbacks.

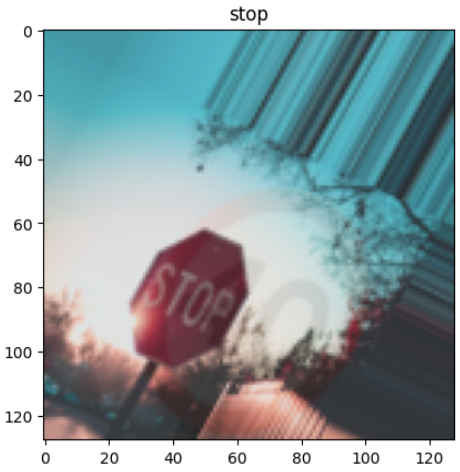
Augmentation de base de données

Utiliser des ensembles de données de taille réduite peut induire un surapprentissage. Afin d'augmenter la variabilité des données d'entraînement et d'améliorer la généralisation du modèle, nous avons utilisé des techniques d'augmentation de données. Ces techniques consistent à appliquer des transformations contrôlées aux images d'entraînement, telles que la rotation, le décalage horizontal et vertical, le cisaillement, le zoom et le retournement horizontal. Cela permet d'augmenter la taille effective de l'ensemble de données et d'améliorer la robustesse du modèle car un réseau de neurones ne sait pas reconnaître des formes dans une image qui sont transformées par translation, rotation ou changement d'échelle.

(a) Exemple 1



(b) Exemple 2.



(c) Exemple 3

(d) Exemple 4

Callbacks

En plus de l'augmentation des données, nous avons intégré des callbacks dans le processus d'entraînement de notre modèle CNN. Les callbacks sont des fonctions appelées à des étapes spécifiques pendant l'entraînement du modèle, telles que la fin d'une epoch ou la diminution de la perte sur l'ensemble de validation.

Nous avons utilisé deux callbacks principaux :

- **Early Stopping:** Ce callback surveille la perte sur l'ensemble de validation et arrête l'entraînement si la perte ne diminue pas pendant un certain nombre d'époques consécutives, ce qui évite le surapprentissage et permet d'économiser du temps de calcul.
- **Model Checkpoint:** Ce callback permet de sauvegarder les poids du modèle à chaque fois que la performance sur l'ensemble de validation s'améliore. Ainsi, à la fin de l'entraînement, nous disposons des poids du modèle qui ont donné les meilleures performances sur l'ensemble de validation.

Nous avons augmenté le nombre d'époques d'entraînement et ajouté des callbacks pour garantir une meilleure gestion de l'entraînement du modèle, ce qui nous permet d'obtenir des résultats plus fiables et généralisables.

3.2.3 Version 3

Architecture détaillée du réseau :

Pour cette dernière étape, nous avons choisi d'utiliser la méthode de fine-tuning vue en cours. Pour la base convolutive, nous avons utilisé la base convolutive du réseau VGG16 déjà pré-entraîné sur la base de données ImageNet, qui contient environ 14 millions d'images. Ce réseau agit comme extracteur de caractéristiques, et nous avons ajouté deux couches denses à la sortie : la première couche contient 256 neurones suivie d'une couche de dropout avec un taux de 0.5, et la dernière couche contient 5 neurones. Dans un premier temps, nous avons gelé les paramètres de l'extracteur de caractéristiques et entraîné uniquement les deux dernières couches du classifieur. Une fois ces couches entraînées, nous avons débloqué les paramètres de la base convolutive et ré-entraîné l'ensemble du réseau pour le spécifier à la nouvelle tâche. Nous avons utilisé un taux d'apprentissage de 1×10^{-5} afin de ne pas risquer de détruire les filtres généraux obtenus lors du pré-entraînement. Enfin, nous avons introduit l'augmentation de données et les mêmes callbacks utilisés lors de l'étape précédente, et nous avons choisi d'entraîner le modèle sur 10 époques.

Résumé de l'architecture :

- Couches convolutionnelles : la base convolutive du réseau VGG16
- Couche entièrement connectée : 1 couche avec 256 neurones plus couche de sortie de 5 neurones
- Taux de dropout : 0.5 après la couche entièrement connectée

Liste des hyperparamètres :

- Taux de dropout : 0.5
- Taux d'apprentissage : Pour le premier entraînement, nous avons utilisé le taux d'apprentissage 3×10^{-4} et pour la dernière étape, nous avons utilisé 1×10^{-5}
- Taille du batch : Default
- Nombre d'époques : 10

Ces différentes étapes nous ont permis de construire et d'améliorer progressivement notre modèle de classification des panneaux de signalisation. En combinant la construction d'un modèle CNN de base, l'augmentation des données et le fine-tuning avec un modèle pré-entraîné, nous avons pu obtenir des performances satisfaisantes dans notre tâche de classification.

4 Analyse des Résultats

4.1 Explication du Recall et du F1-score

4.1.1 Recall (Rappel)

Le *recall*, ou rappel, est une métrique utilisée pour évaluer la capacité d'un modèle à identifier correctement tous les exemples pertinents dans un ensemble de données. En d'autres termes, il mesure la proportion de vrais positifs parmi tous les exemples pertinents (vrais positifs + faux négatifs).

$$\text{Recall} = \frac{\text{Vrais Positifs (TP)}}{\text{Vrais Positifs (TP)} + \text{Faux Négatifs (FN)}} \tag{1}$$

Interprétation :

- Un recall élevé signifie que le modèle réussit à identifier la plupart des exemples pertinents (il y a peu de faux négatifs).
- Un recall faible indique que le modèle manque beaucoup d'exemples pertinents (il y a beaucoup de faux négatifs).

4.1.2 F1-score

Le *F1-score* est la moyenne harmonique de la précision (*precision*) et du recall. Il est utilisé pour trouver un équilibre entre les deux métriques.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

Interprétation :

- Le F1-score prend en compte à la fois la précision et le recall. Il est utile lorsque on a besoin de maintenir un équilibre entre ces deux mesures.
- Un F1-score élevé indique que le modèle a un bon équilibre entre précision et rappel.

4.1.3 Exemples

Précision (Precision) : Proportion de vrais positifs parmi tous les positifs prévus (vrais positifs + faux positifs).

$$\text{Precision} = \frac{\text{Vrais Positifs (TP)}}{\text{Vrais Positifs (TP)} + \text{Faux Positifs (FP)}} \tag{3}$$

Recall (Rappel) : Proportion de vrais positifs parmi tous les exemples pertinents (vrais positifs + faux négatifs).

F1-score : Combine la précision et le recall en une seule métrique.

4.2 Model 1

- **Test loss :** 0.8100
- **Test accuracy :** 0.7935

Rapport de Classification :

- **Classe 0 :** Precision = 0.89, Recall = 0.77, F1-score = 0.83
- **Classe 1 :** Precision = 0.90, Recall = 0.87, F1-score = 0.89
- **Classe 2 :** Precision = 0.67, Recall = 0.84, F1-score = 0.74
- **Classe 3 :** Precision = 0.82, Recall = 0.87, F1-score = 0.84
- **Classe 4 :** Precision = 0.73, Recall = 0.61, F1-score = 0.67

Analyse :

- Le modèle a une précision globale de 79.35%, ce qui montre une performance acceptable mais avec des marges de progrès.
- Les classes 'crosswalk' et 'no-entry' ont des précisions et des recalls relativement élevés, indiquant une bonne performance de classification pour ces classes.
- La classe 'speedlimit' montre une faiblesse en précision (0.67) mais un bon recall (0.84), suggérant que le modèle identifie correctement la plupart des exemples de cette classe mais avec un certain nombre de faux positifs.
- La classe 'trafficlight' a des scores plus faibles, en particulier un recall de 0.61, indiquant que le modèle a du mal à identifier correctement cette classe.

4.3 Model 2

- **Test loss :** 0.4619
- **Test accuracy :** 0.8452

Rapport de Classification :

- **Classe 0 :** Precision = 0.88, Recall = 0.90, F1-score = 0.89

- **Classe 1 :** Precision = 0.82, Recall = 0.90, F1-score = 0.86
- **Classe 2 :** Precision = 0.73, Recall = 0.87, F1-score = 0.79
- **Classe 3 :** Precision = 0.93, Recall = 0.90, F1-score = 0.92
- **Classe 4 :** Precision = 0.91, Recall = 0.65, F1-score = 0.75

Analyse :

- Le modèle 2 montre une amélioration notable par rapport au modèle 1 avec une précision globale de 84.52%.
- Les classes 'crosswalk', 'no-entry' et 'stop' ont de très bons résultats en précision et recall, ce qui indique une meilleure capacité du modèle à identifier ces classes correctement.
- La classe 'speedlimit' voit une amélioration du F1-score à 0.79, ce qui montre une meilleure balance entre précision et recall.
- La classe 'trafficlight' reste la plus difficile à classifier correctement, bien que la précision soit élevée (0.91), le recall est encore relativement bas (0.65).

4.4 Model 3

- **Test loss :** 0.2473
- **Test accuracy :** 0.9290

Rapport de Classification :

- **Classe 0 :** Precision = 0.90, Recall = 0.90, F1-score = 0.90
- **Classe 1 :** Precision = 1.00, Recall = 0.97, F1-score = 0.98
- **Classe 2 :** Precision = 0.88, Recall = 0.97, F1-score = 0.92
- **Classe 3 :** Precision = 0.97, Recall = 0.94, F1-score = 0.95
- **Classe 4 :** Precision = 0.90, Recall = 0.87, F1-score = 0.89

Analyse :

- Le modèle 3 a les meilleures performances globales avec une précision de 92.90%, ce qui représente une amélioration substantielle par rapport aux modèles précédents.
- Les classes 'no-entry', 'speedlimit', et 'stop' montrent des scores F1 très élevés, avec une précision et un recall presque parfaits pour la classe 1.
- La classe 'trafficlight' a également montré une nette amélioration avec un F1-score de 0.89, bien que le recall de 0.87 indique encore des marges de progrès.
- Globalement, le modèle 'stop' démontre une excellente capacité à classifier correctement les panneaux de signalisation avec des scores équilibrés entre les différentes classes.

Conclusion :

- **Model 1** est une base acceptable, mais montre des faiblesses notables dans la classification de certaines classes, en particulier la classe 'trafficlight' et présente un sur-apprentissage.
- **Model 2** apporte une amélioration significative avec des techniques d'augmentation des données et des callbacks, mais encore quelques défis dans la classification de la classe 'trafficlight'.
- **Model 3** offre la meilleure performance, probablement en raison de l'utilisation du fine-tuning avec un modèle pré-entraîné comme VGG16, montrant des scores élevés et bien équilibrés pour toutes les classes, et une réduction significative de la loss.

4.5 Matrice de Confusion

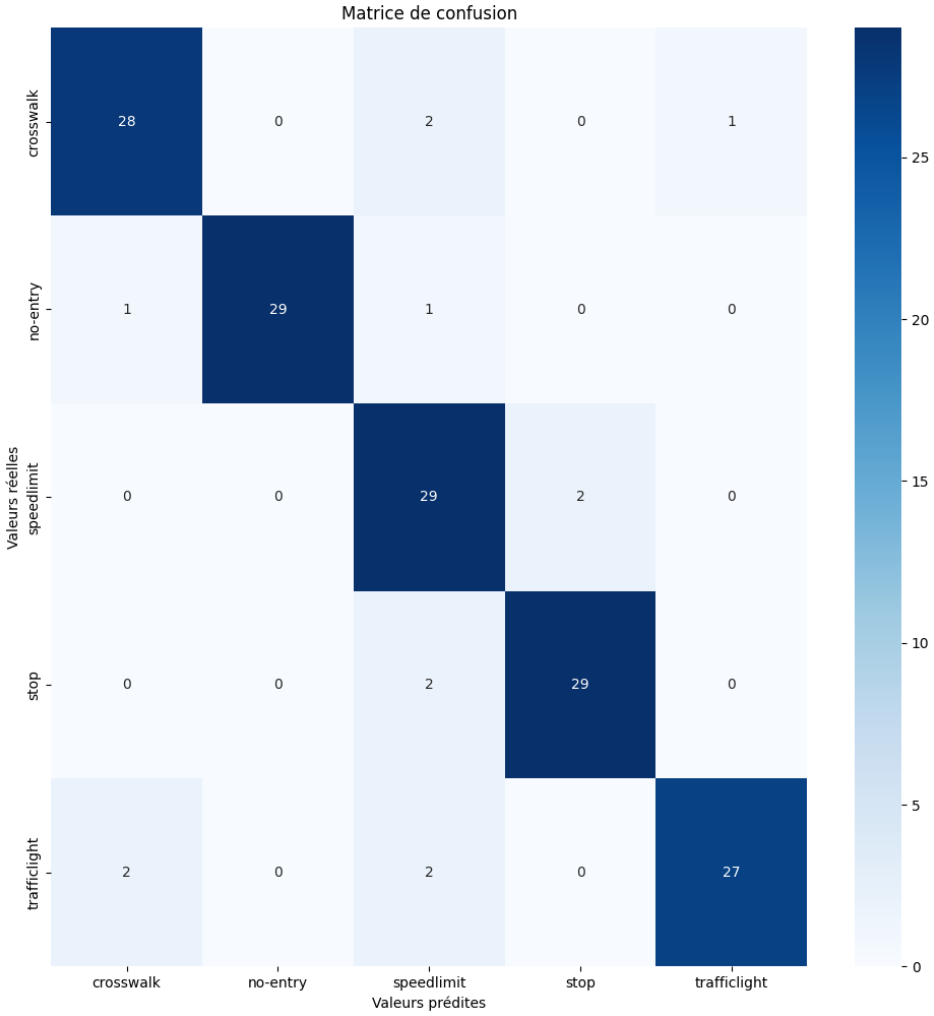


Figure 11: Matrice de Confusion

On peut voir dans la figure suivante une partie des exemples que notre modèle avait du mal à prédire:

(a) Exemple 1 (b) Exemple 2 (c) Exemple 3

(d) Exemple 4 (e) Exemple 5 (f) Exemple 6

Figure 12: Grille de 9 exemples

- Dans le premier exemple, le modèle a prédit un feu tricolore au lieu d'un panneau stop. Cela peut être expliqué par la présence simultanée des deux éléments dans l'image.
- Dans la plupart des photos de feux tricolores, l'environnement comprend souvent un poteau. Les images dans les exemples 2, 5 et 8 montrent des poteaux similaires à ceux des feux tricolores, ce qui a probablement induit le modèle en erreur.
- Dans l'exemple 3, il n'y a pas de raison apparente pour que le modèle identifie l'image comme un panneau d'interdiction. Cela pourrait être dû à la présence d'un bâtiment en arrière-plan, similaire à ceux présents derrière les panneaux d'interdiction dans l'ensemble d'entraînement.
- Dans l'exemple 4, la photo contient à la fois un feu tricolore et un panneau qui ressemble à un passage pour piétons. Le modèle a donc prédit que c'était un passage piéton plutôt qu'un feu tricolore en raison de cette confusion visuelle.

4.6 Discussion

Bilan de performance Synthèse des points forts : Modèle robuste et performant, bonne généralisation grâce à l'augmentation des données, "call-backs" et Fine-tuning.

Identification des principales faiblesses : Erreurs dues aux conditions d'éclairage difficiles, obstructions ou angles inhabituels.

Perspectives d'Amélioration Améliorations de la base de données : Ajout de données supplémentaires, amélioration de la qualité des annotations, techniques de data augmentation avancées.

Améliorations du modèle : Exploration de nouvelles architectures, techniques de régularisation robustes, optimisation des hyperparamètres via des méthodes sophistiquées.

Intégration des ROI dans l'Architecture du Modèle : Pour intégrer des ROI dans notre modèle, nous pourrions explorer des architectures telles que R-CNN. Ces architectures combinent des réseaux neuronaux convolutifs avec des mécanismes de détection d'objets pour identifier et extraire automatiquement les régions d'intérêt dans les images.

5 Conclusion

Ce projet de classification d'images a été une expérience enrichissante, nous permettant de mettre en pratique diverses techniques de deep learning pour résoudre une tâche complexe. Voici un bilan de notre travail et une réflexion sur les perspectives d'amélioration :

5.1 Bilan du Travail

Exploration Approfondie des Modèles : Nous avons commencé par construire un modèle CNN classique, puis nous avons progressivement évolué vers des architectures plus complexes telles que le fine-tuning avec VGG16, en intégrant des techniques avancées comme l'augmentation des données et l'utilisation de callbacks pour l'early stopping et le sauvegarde des meilleurs modèles.

Pour améliorer le projet, nous pourrions envisager d'explorer davantage les architectures de réseaux neuronaux, d'investiguer des techniques de régularisation plus robustes et d'optimiser les hyperparamètres en utilisant des méthodes plus sophistiquées. De plus, l'ajout de données supplémentaires et l'amélioration de la qualité des annotations pourraient renforcer la base de données. Nous pourrions également envisager d'incorporer des techniques avancées de data augmentation pour augmenter la diversité des données d'entraînement. En outre, l'intégration de modèles exploitant les régions d'intérêt (ROI) pourrait potentiellement améliorer la performance du système, en tenant compte des zones spécifiques des images les plus pertinentes pour la tâche de classification.

Les difficultés rencontrées comprenaient la compréhension des résultats du modèle, la collecte et l'annotation des données, la nécessité d'une compréhension spécialisée de CNNs

pour extraire des caractéristiques pertinentes, ainsi que la gestion des priorités pour maintenir un équilibre entre la qualité du travail et les délais.