PROJET DEEP LEARNING

Mohamed Kallel, Simon-Pierre Rodner, Jean-Christophe Rigoni

1. Introduction

Contexte et Motivation :

C’est principalement pour le développement des véhicules autonomes que les systèmes de reconnaissance des panneaux de signalisation routière ont bénéficié d’un intérêt croissant. Notre projet vise à réaliser un modèle basé sur des réseaux de neurones convolutifs (CNN) dont le but est de reconnaître les panneaux de signalisation présents dans les images du dataset GTSRB (German Traffic Sign Recognition Benchmark). Le succès de l’algorithme est crucial pour assurer la sécurité dans la prise de décision automatique en conduite autonome.

Objectif du projet :

Notre objectif est de construire, entraîner et évaluer un modèle CNN performant pour la classification des panneaux de signalisation, en expérimentant avec différentes architectures, augmentations de données et techniques de régularisation pour aboutir à un modèle robuste.

2. Exploration du Dataset

Présentation du Dataset GTSRB :

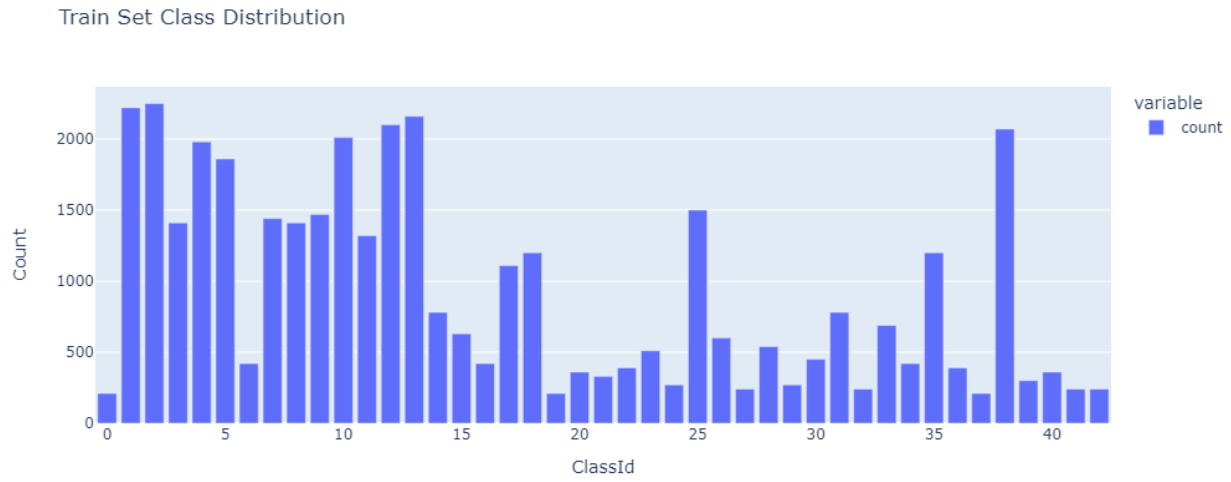
Le dataset **GTSRB** contient plus de 50K images de panneaux de signalisation répartis en **43 classes**. Chaque classe représente un type de panneau spécifique, allant des limitations de vitesse aux panneaux de danger ou d'indication.

Le dataset est séparé en deux dossiers, Train avec 37K images répartis en 43 sous dossiers, et Test avec 12K images. Toutes les classes sont représentées dans les deux datasets.

Exploration des données :

Nous avons commencé par visualiser quelques images du dataset pour comprendre la diversité des panneaux et des conditions d'éclairage, de qualité, etc. Nous constatons une grande disparité de taille, de qualité, et de luminosité des images. Le plus grand défaut du dataset est que chaque dossier de classe d'image est en fait constitué de très peu de photos copiées multiples fois avec un léger changement comme un zoom, ou un décalage. Finalement, le train dataset est constitué de très peu d'images originales.

D'autres part, la distribution des classes est déséquilibrée; certaines classes ont dix fois plus d'exemples que d'autres. Néanmoins ce ne sont pas des rapports exponentiels, et la robustesse du modèle et les augmentations appliquées ont réussi à effacer ce problème.



3. Préparation des Données

Prétraitement des images :

Pour normaliser la taille des images en entrée du modèle, nous avons redimensionné chaque image à **64x64 pixels** par interpolation bilinéaire, choix par défaut qui est un bon compromis entre qualité et vitesse. Le travail de normalisation (division de la valeur des pixels par 255 pour les ramener entre 0 et 1) est réservé au premier layer du modèle.

Encodage des labels:

Nous avons encodé sous forme de one hot encoder les 43 labels avec la fonction to_categorical pour pouvoir utiliser la fonction de coût catégorical_crossentropy.

Taille des Batchs

Nous avons choisi des batchs de 64 images pour optimiser le temps de calcul en gardant de la précision.

Augmentation des données :

Pour améliorer la robustesse du modèle et éviter le surapprentissage, nous avons appliqué plusieurs techniques d'**augmentation des données** :

- **Rotation** aléatoire.
- **Translation** horizontale et verticale.
- **Zoom**.
- **Variations de luminosité**.

Ces techniques permettent d'augmenter artificiellement la taille du dataset et de rendre le modèle plus généraliste.

Division des données :

Les données du dossier Train ont été divisées en deux ensembles :

- **Entraînement** (80 % des images).
- **Validation** (20 %).

Les images du dossier Test ont été utilisées intégralement pour l'évaluation du modèle.

4. Conception du Modèle CNN

Architecture du modèle :

Nous avons conçu un modèle CNN en plusieurs étapes :

1. **4 Couches de convolution** : Extraction des caractéristiques locales des images avec des filtres 3x3, suivies de couches de **MaxPooling** pour réduire la dimensionnalité et un nombre croissant de filtres pour étudier les détails des images de plusieurs manières. Le nombre de filtres de chaque nouvelle couche double par rapport à la précédente.
2. **1 Couche fully connected** : Après avoir aplati les caractéristiques extraites, nous avons ajouté une seule couche entièrement connectée pour interpréter les caractéristiques et prédire les classes.
3. **Régularisation** : Nous avons utilisé **Dropout** et **Batch Normalization** pour améliorer la généralisation du modèle et éviter le surapprentissage.

Architecture finale :

- **1 Couche Rescaling**
- **4 Couches Convolutionnelles + Pooling + BatchNormalisation**
- **1 Couche Flatten**
- **1 Dense layer** (256)
- **1 Dropout** (0.5)
- **1 Dense Layer** (43) pour la classification

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 64, 64, 3)	0
conv2d_4 (Conv2D)	(None, 62, 62, 16)	448
max_pooling2d_3 (MaxPooling2D)	(None, 31, 31, 16)	0
batch_normalization_4 (BatchNormalization)	(None, 31, 31, 16)	64
conv2d_5 (Conv2D)	(None, 29, 29, 32)	4,640
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 32)	0
batch_normalization_5 (BatchNormalization)	(None, 14, 14, 32)	128
conv2d_6 (Conv2D)	(None, 12, 12, 64)	18,496
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 64)	0
batch_normalization_6 (BatchNormalization)	(None, 6, 6, 64)	256
conv2d_7 (Conv2D)	(None, 4, 4, 128)	73,856
max_pooling2d_6 (MaxPooling2D)	(None, 2, 2, 128)	0
batch_normalization_7 (BatchNormalization)	(None, 2, 2, 128)	512
flatten_1 (Flatten)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131,328
dropout_1 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 43)	11,051

5. Entraînement du Modèle

Fonction de coût et Optimiseur :

Nous avons utilisé la fonction de coût **catégorical cross entropy** adaptée à la classification multi-classes avec des label sous forme “one hot encoding”.

L’optimiseur **Adam** a été sélectionné pour son efficacité en termes de convergence rapide. Nous avons également utilisé un **annealing du taux d'apprentissage** pour réduire progressivement ce dernier et stabiliser la convergence: ReduceLROnPlateau

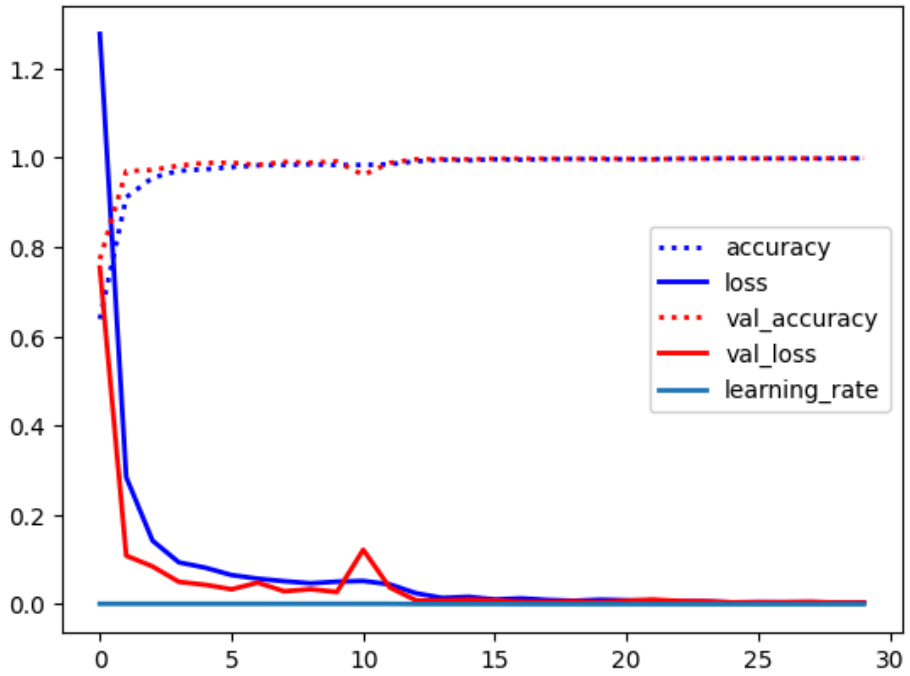
Régularisation et callbacks :

Pour monitorer l'entraînement et éviter l'overfitting :

- **EarlyStopping** pour arrêter l'entraînement lorsque la validation n'améliore plus.
- **ReduceLROnPlateau** pour ajuster dynamiquement le taux d'apprentissage.

Courbes de performance :

Les courbes de perte et d’accuracy montrent une bonne convergence avec un modèle atteignant une **accuracy** et une **val accuracy** de près de **99 %** sur l'ensemble d'entraînement.



6. Évaluation du Modèle

Métriques de Performance :

Sur l'ensemble de test, le modèle a atteint une précision de **98 %**. Les **métriques de performance** calculées incluent :

- **Accuracy**.
- **Recall**.
- **F1-score**.

Matrice de confusion : Une matrice de confusion a été générée pour analyser les erreurs. Les erreurs proviennent principalement de panneaux très similaires comme certains panneaux triangulaires indicateurs de dangers. Certaines figures ont des caractéristiques très similaires comme la double courbe et l'animal bondissant, ou l'intersection et le flocon de neige.

Exemple d'erreurs de classifications du modèle:

7. Améliorations et Expérimentations

Ajustement des hyperparamètres :

Différents hyperparamètres ont été ajustés pour améliorer les performances :

- **Taille des batchs** : Nous avons testé plusieurs tailles (32, 64, 128) pour trouver la plus performante (64)
- **Taux d'apprentissage** : Un ajustement fin a été réalisé avec **ReduceLROnPlateau** ou nous avons testé plusieurs factor, patience et learning rate minimum.

Transfer Learning :

Pour améliorer davantage les performances, nous avons expérimenté avec un modèle **pré-entraîné sur ImageNet** en fine-tuning sur GTSRB. Le modèle ResNet-50, par exemple, a légèrement amélioré les performances sur les classes les plus complexes.

Amelioration du dataset :

Obtenir une plus grande quantité de photos et de meilleures qualités.

8. Visualisation des Résultats

- **Visualisation des activations :**

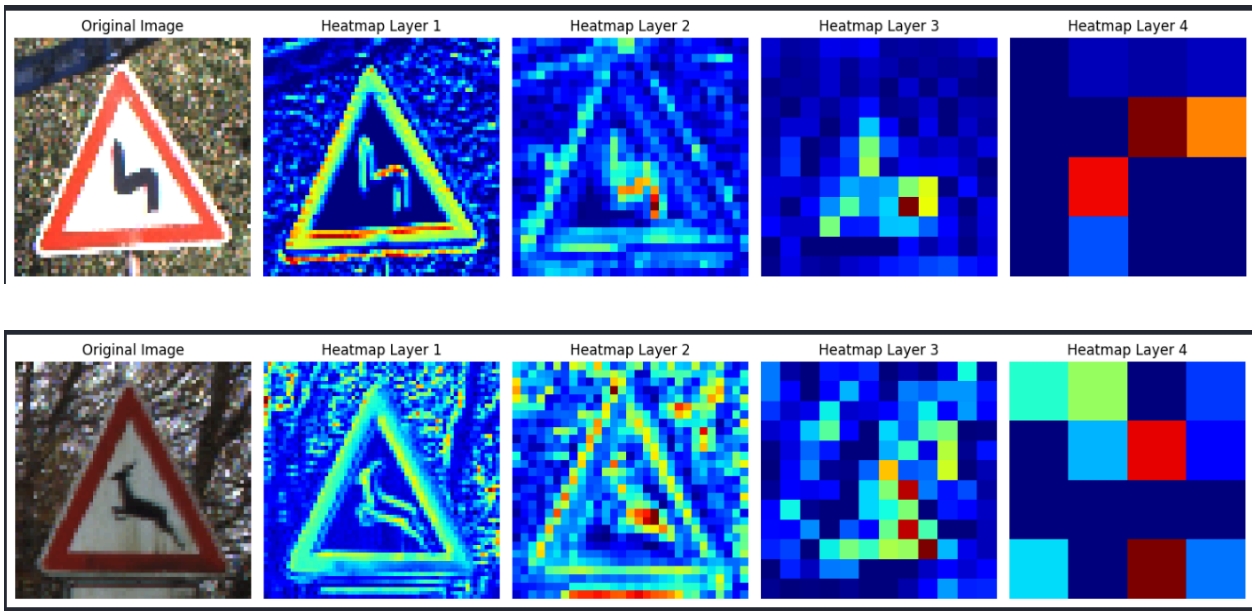
Pour mieux comprendre comment le modèle prend ses décisions, nous avons visualisé les **activations des différentes couches** du CNN. Cela nous a permis de voir comment les couches convolutives détectent des motifs de plus en plus complexes à travers les différentes couches.

- **Grad-CAM :**

Nous avons implémenté **Grad-CAM** pour visualiser quelles parties des images sont utilisées par le modèle pour prendre ses décisions. Ces visualisations ont montré que le modèle se concentre correctement sur les panneaux dans les images, mais parfois se trompe en raison de perturbations dans l'image.

- **Analyse des erreurs :**

Les erreurs les plus courantes proviennent de la **confusion entre des classes très similaires**, comme les panneaux triangulaires avertissant d'un danger contenant des figures dont la forme générale est confondante comme la double courbe et l'animal sauvage ci-dessous.



9. Conclusion

Ce projet a permis de développer un modèle CNN performant pour la classification des panneaux de signalisation routière. En combinant des techniques d'augmentation des données, de régularisation, et d'amélioration des hyperparamètres, nous avons obtenu des résultats compétitifs, en essayant de concevoir un modèle relativement simple et compréhensible. Des améliorations futures pourraient inclure l'utilisation de réseaux plus profonds ou des techniques avancées comme l'intégration reconnaissance de caractères (comme pour les panneaux de limitation de vitesse).