TP2 : Premier Réseau de neurones artificiels

1st Picard Hugo
Intelligence artificielle embarquée
ESIEE – 2025
France
hugo.picard@edu.esiee.fr

2nd Saint-Germain Teddy
Intelligence artificielle embarquée
ESIEE – 2025
France
teddy.saint-germain@edu.esiee.fr

Résumé : Dans ce TP, nous étudions la classification des panneaux de signalisation du dataset GTSRB en entraînant plusieurs architectures de réseaux de neurones, des modèles les plus simples aux premiers CNN. L'objectif est de comprendre comment ces différentes structures traitent l'image, quelles performances elles atteignent et pourquoi certaines approches fonctionnent mieux que d'autres. Ce travail sert surtout à montrer, étape par étape, comment un réseau apprend réellement à reconnaître un panneau routier.

I - Introduction

Dans ce TP, l'idée est assez simple : essayer de comprendre comment un réseau de neurones apprend à reconnaître des panneaux de signalisation à partir d'images. On s'appuie pour ça sur le dataset GTSRB, qui rassemble des photos réelles de panneaux allemands, parfois nettes, parfois floues, parfois prises sous des angles un peu bancals. Bref, un bon terrain de jeu pour voir ce qu'un modèle arrive réellement à comprendre.

On part de modèles très basiques, qui ne voient l'image que comme une longue liste de pixels, puis on ajoute progressivement des convolutions pour leur permettre de repérer des formes, des contours, des motifs. À chaque étape, on observe comment la structure du réseau change sa façon de "regarder" l'image, et surtout comment cela se traduit en performance.

L'objectif n'est pas seulement de trouver le meilleur score, mais de suivre cette montée en compétences du modèle et de comprendre pourquoi certaines architectures fonctionnent mieux que d'autres.

Pour comprendre cette progression, nous allons d'abord présenter les différentes architectures mises en place dans le TP, puis analyser leurs performances avant de comparer ce que chaque modèle parvient réellement à apprendre de ces images de panneaux.

II - Rappels théoriques

Avant d'analyser les modèles entraînés dans ce TP, il est utile de rappeler rapidement comment un réseau de neurones traite une image et ce qui différencie un modèle dense d'un modèle convolutionnel. Ces notions permettent de comprendre pourquoi certaines architectures fonctionnent mieux que d'autres sur un dataset visuel comme GTSRB.

Un réseau dense classique relie tous les neurones entre eux : chaque pixel est connecté à chaque neurone des couches suivantes. Sur le papier, cela donne un modèle flexible, mais dans les faits, il ne comprend rien à la structure d'une image. Il traite tous les pixels comme des valeurs indépendantes, ce qui l'empêche de capter les formes, les bords ou la position relative des éléments.

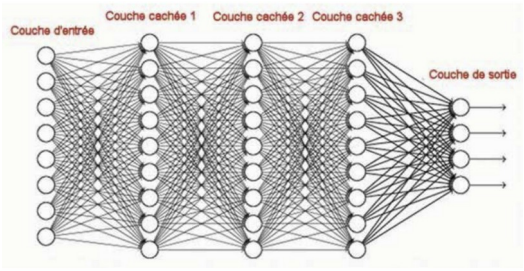

Figure 1 – réseau dense

Les réseaux convolutionnels[1] ont été conçus pour pallier cette limite. Une convolution applique un petit filtre (par exemple 3×3 ou 5×5) qui glisse sur l'image et repère des motifs locaux. Cette opération permet au réseau d'apprendre des patterns visuels simples, comme des lignes, des angles ou des textures. En empilant plusieurs convolutions, le modèle apprend progressivement des structures de plus en plus complexes.

[1] https://laurentnajman.org/uploads/images/DeepLearning/ConvNets.pdf

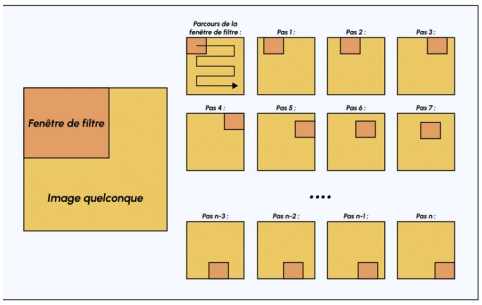

Figure 2 – Convolutions[1]

Le MaxPooling est souvent associé aux convolutions. Il réduit la taille des cartes de caractéristiques en ne conservant que les valeurs les plus importantes dans chaque région. Cette opération permet au réseau d'être moins sensible à de petites variations (bruit, translations) et réduit le nombre de paramètres, ce qui stabilise l'apprentissage.

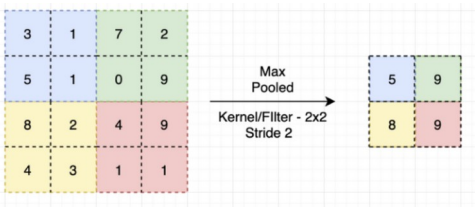

Figure 3 - MaxPooling

Lorsque l'image a été transformée en caractéristiques visuelles, il faut passer à la phase de classification. La couche Flatten convertit simplement les cartes 2D obtenues par les convolutions en un long vecteur. Ce vecteur peut alors être traité par des couches pleinement connectées, qui combinent l'information pour produire une prédiction finale.

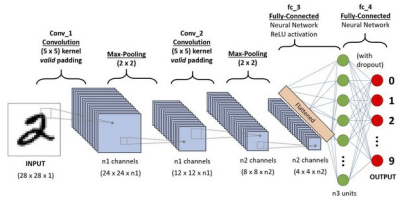

Figure 4 – Convolution + MaxPooling + Flatte

Enfin, certains modèles utilisent du Dropout, une méthode de régularisation qui désactive aléatoirement une partie des neurones pendant l'entraînement. Cela empêche le réseau de s'appuyer trop fortement sur un petit nombre de connexions et améliore sa capacité à généraliser sur de nouvelles images, ce qui est particulièrement utile dans un dataset varié comme GTSRB.

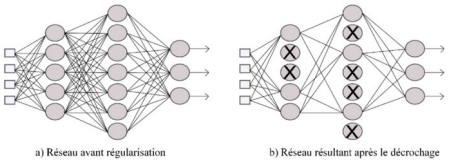

Figure 5 - Dropout

Ces éléments théoriques expliquent pourquoi les modèles du TP ne présentent pas les mêmes performances. Selon qu'ils utilisent une simple succession de couches denses ou qu'ils exploitent les convolutions, qu'ils régularisent l'apprentissage ou non, leur capacité à comprendre l'image et à faire des prédictions fiables peut varier de manière très nette.

III - Description des modèles

Les modèles que nous avons construits suivent une logique simple : commencer par quelque chose de basique, puis ajouter progressivement des briques plus adaptées à la vision. L'objectif n'est pas seulement de comparer leurs performances, mais de comprendre ce que chaque étape apporte vraiment lorsque l'on travaille sur des images comme celles du dataset GTSRB.

Le premier modèle, (notebook: model), est volontairement minimaliste. Il se contente d'aplatir l'image pour la transformer en un long vecteur, puis de passer ce vecteur dans deux couches entièrement connectées. Ce type de réseau ne voit pas l'image comme une image. Il traite les pixels comme une succession de valeurs indépendantes, sans notion de forme ou de proximité. V1 permet donc de poser une base : qu'est-ce qu'un réseau peut comprendre s'il ne s'appuie sur aucune structure visuelle ?

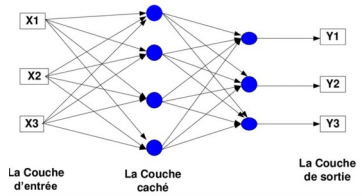

Figure 6 - model

Le modèle V2, (notebook: model_v2), reprend exactement le même principe, mais avec une couche dense supplémentaire. Ce n'est pas un changement radical, mais cela augmente la capacité du réseau à apprendre des relations un peu plus complexes dans les données. En pratique, V2 doit logiquement mieux performer que V1, mais il reste limité pour les mêmes raisons : il ignore complètement l'organisation spatiale de l'image.

Le modèle V3 introduit enfin ce qui manque cruellement aux deux premiers : une couche convolutive. Dès qu'on ajoute une convolution, le réseau arrête de regarder les pixels isolément. Il commence à détecter des motifs locaux, comme des bords, des coins ou des zones de contraste, ce qui est exactement ce qui caractérise un panneau de signalisation. Cette simple modification change profondément les représentations internes du modèle. Après la convolution, le signal passe en vecteur puis dans deux couches denses pour la classification.

C'est le premier modèle de la série qui « regarde » vraiment l'image.

Le modèle V4 va un peu plus loin dans cette direction. On enchaîne plusieurs convolutions, ce qui permet d'extraire des motifs de plus en plus abstraits. On ajoute également du Dropout dans les couches denses finales pour éviter que le modèle ne s'habitue trop à la base d'entraînement et pour l'obliger à généraliser. V4 ressemble davantage à un petit CNN tel qu'on les utilise en pratique : il est plus profond, plus robuste et mieux armé pour traiter des images variées comme celles du GTSRB.

Cette progression du modèle le plus naïf au modèle « le plus structuré » permet de voir, quasiment sous nos yeux, comment un réseau change de comportement dès qu'on lui donne les bons outils pour comprendre une image. C'est exactement ce qu'on cherche à montrer dans ce notebook

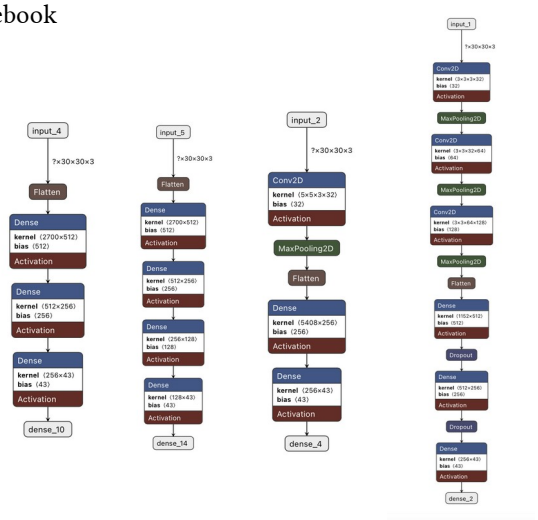

Figure 7 – Représentation des modèles sur Netron

IV - Résultats expérimentaux

L'entraînement des différents modèles met en évidence des comportements assez contrastés, qui reflètent directement leur architecture. Même si tous ont été entraînés dans les mêmes conditions, sur le même jeu d'images et avec les mêmes paramètres, leurs courbes d'apprentissage montrent que certains réseaux comprennent beaucoup mieux la structure visuelle du dataset GTSRB que d'autres.

Quand on regarde les deux premiers modèles (V1 et V2), on voit tout de suite leurs limites. Ce sont des réseaux entièrement denses qui commencent par aplatir l'image, donc ils ne tiennent pas compte de sa structure. Résultat : ils apprennent très vite le jeu d'entraînement et montent presque à 100 % d'accuracy, mais la validation reste en dessous. La loss d'entraînement tombe quasiment à zéro alors que la loss de validation stagne plus haut. En clair, ils mémorisent les images au lieu de vraiment comprendre ce qu'elles contiennent. Ça fonctionne, mais ça ne généralise pas très bien.

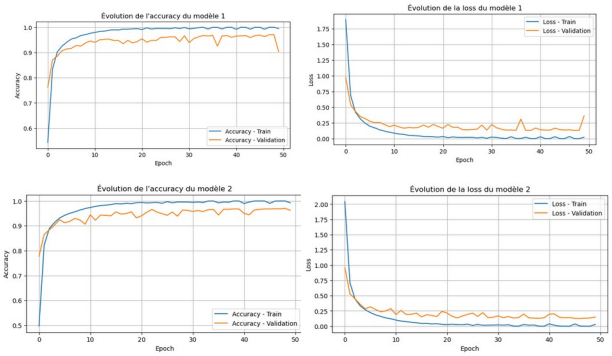

Figure 8 – Courbes d'évolutions V1-V2

Avec le modèle V3, on change déjà de catégorie. Le simple fait d'ajouter une couche de convolution avant le Flatten permet au réseau de capter des formes et des motifs, et ça se voit dans les courbes : train et validation se rapprochent, l'accuracy de validation grimpe autour de 98–99 %, et la loss reste stable. Le modèle commence vraiment à "voir" les panneaux, pas juste à retenir des pixels.

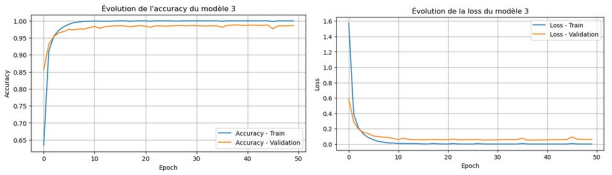

Figure 9 – Courbe d'évolution v3

Le modèle V4, lui, pousse vraiment la logique jusqu'au bout. Trois blocs convolution + pooling lui permettent d'apprendre des représentations de plus en plus fines, et le Dropout limite le surapprentissage. Les courbes sont propres, les performances très hautes et surtout très stables : l'écart entre train et validation est minime, la loss validation reste basse, rien ne s'écroule en fin d'apprentissage. On sent un modèle qui a trouvé un bon équilibre.

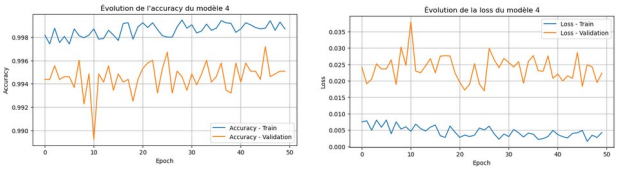

Figure 10 – Courbe d'évolution v3

Au final, V4 est clairement le meilleur choix. Il extrait mieux les informations visuelles, il généralise mieux, et sa stabilité montre qu'il n'est pas juste en train de mémoriser. C'est l'architecture la plus adaptée à une vraie tâche de vision comme la reconnaissance de panneaux.

Les matrices de confusion normalisées renforcent ce qu'on observait déjà sur les courbes. Sur le modèle V1, la diagonale est bien visible mais reste imparfaite : plusieurs classes montrent encore des erreurs notables, avec des pixels jaunes ou orangés qui trahissent des confusions récurrentes entre panneaux visuellement proches. Comme le réseau n'utilise que des couches denses, il manque de capacité à extraire des motifs visuels précis, ce qui l'amène à se tromper ponctuellement sur des panneaux qui partagent des formes ou des couleurs similaires.

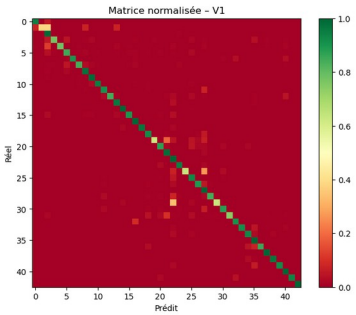

Figure 11 – Matrice de confusion V1

Avec le modèle V2, la diagonale devient plus nette et les erreurs se réduisent, mais on reste dans la même logique que V1 : quelques classes sont bien reconnues, d'autres un peu moins, et les erreurs se dispersent un peu partout. On voit bien que l'ajout d'une couche dense supplémentaire améliore légèrement la situation, mais pas suffisamment pour capturer la structure réelle des panneaux. Le réseau continue d'apprendre essentiellement des corrélations globales sur les pixels, ce qui limite sa précision sur certaines catégories.

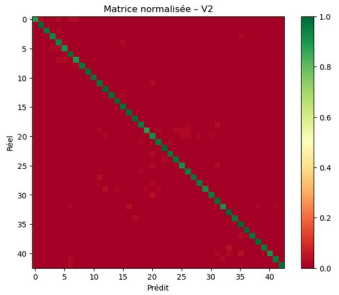

Figure 12 – Matrice de confusion V2

Le modèle V3 apporte un vrai saut qualitatif. L'introduction d'une unique couche convolutionnelle suffit à clarifier la matrice de confusion : la diagonale est beaucoup plus marquée, les erreurs sont beaucoup plus rares et mieux localisées. Le réseau commence clairement à comprendre les motifs distinctifs des panneaux (formes circulaires, interdictions, limitations de vitesse, etc.). Certaines classes montrent encore de légères confusions, mais globalement la structure de la matrice devient propre, ce qui traduit une meilleure généralisation.

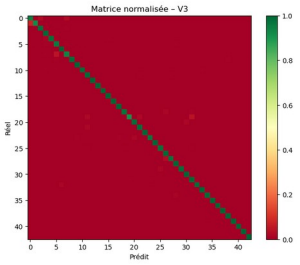

Figure 12 – Matrice de confusion V3

Le modèle V4 présente la matrice la plus nette : quasiment toute la diagonale apparaît en vert foncé, signe que la majorité des classes sont reconnues avec un taux de réussite très élevé. Les erreurs résiduelles sont isolées et peu nombreuses. Les trois blocs convolution + pooling permettent au réseau de distinguer finement les panneaux, même ceux qui se ressemblent beaucoup. Le Dropout, quant à lui, stabilise l'apprentissage et limite les dérives. C'est le modèle qui comprend le mieux la structure visuelle du dataset, et sa matrice de confusion reflète cette maturité.

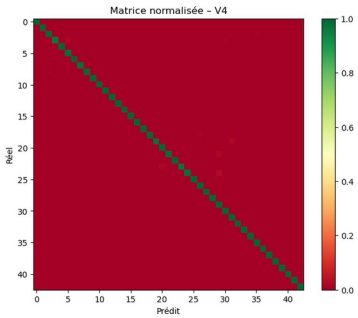

Figure 13 – Matrice de confusion V4

En résumé, les matrices confirment que V4 est le meilleur modèle : c'est celui qui fait le moins d'erreurs, celui qui différencie le mieux les classes proches et celui dont la diagonale est la plus uniforme. Les CNN simples (V3) font déjà un travail solide, mais seule l'architecture plus profonde de V4 permet d'atteindre une compréhension très fine des panneaux et une performance presque « parfaite. »

Pour évaluer la capacité de généralisation des quatre modèles, nous avons aussi utilisé une image test provenant de l'extérieur du dataset d'entraînement. L'idée était de vérifier non seulement leur précision, mais surtout leur aptitude à reconnaître correctement un panneau dans des conditions qui ne correspondent pas exactement aux exemples appris.

Le modèle V4 se distingue immédiatement. Il identifie la classe 33 avec une confiance totale et une distribution des probabilités entièrement concentrée sur la bonne réponse. Cette réaction montre qu'il parvient à extraire une représentation suffisamment riche et abstraite du panneau pour rester fiable même hors distribution. La

reconnaissance ne repose pas sur quelques détails spécifiques, mais sur une compréhension globale de la forme et du pictogramme et confirme donc l'analyse faite juste avant.

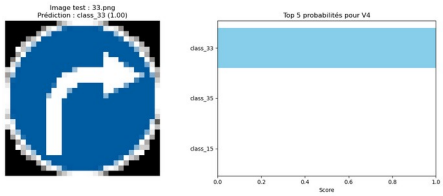

Figure 14 – test avec image

Le modèle V3 arrive au même résultat, lui aussi avec une forte certitude, même si sa confiance est légèrement moins extrême que celle de V4. La prédiction reste nette et cohérente, avec très peu d'ambiguïté. On voit que V3 a déjà atteint un niveau suffisant pour analyser l'image dans son ensemble, mais qu'il reste un peu plus sensible que V4 aux variations de texture ou de couleur.

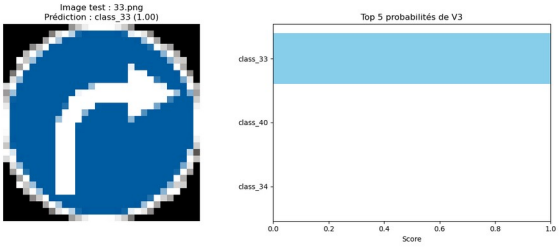

Figure 15 – test avec image v3

Avec V2, le comportement change. Le modèle identifie bien la bonne classe, mais la confiance chute déjà à un niveau plus modéré. D'autres classes commencent à apparaître dans le top des probabilités, signe qu'il hésite dès que l'image ne correspond pas exactement aux exemples rencontrés pendant l'apprentissage. V2 reste dépendant de motifs locaux et manque encore de capacité pour interpréter le panneau de manière globale, ce qui limite sa robustesse.

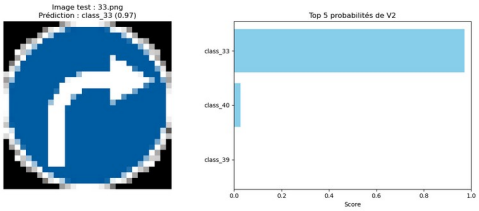

Figure 16 – Test avec image v2

Enfin, V1 confirme ses limites. C'est le seul modèle qui se trompe et prédit la classe 40 avec une certitude faible et très diffuse. La distribution des probabilités est instable, et aucun choix ne s'impose vraiment pour lui. Cette erreur montre clairement que V1 ne parvient pas à extraire les bonnes caractéristiques dès que les conditions changent un peu. Sa perception repose presque exclusivement sur des détails isolés, et non sur la structure du panneau.

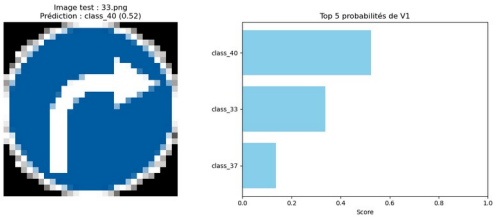

Figure 17 – Test avec image v1

Au final, cette simple image hors dataset met en évidence la progression entre les quatre versions. Plus l'architecture gagne en profondeur et en capacité d'abstraction, plus le modèle devient capable de reconnaître correctement un panneau même lorsque les conditions diffèrent de celles vues à l'entraînement. V4 apparaît comme le premier modèle réellement robuste, V3 comme une version solide mais légèrement moins sûre, V2 comme un intermédiaire encore fragile, et V1 comme une architecture trop limitée pour généraliser correctement.

V - Conclusion

Cette étude montre que le passage d'un petit réseau comme V1 à un modèle plus structuré comme V4 change complètement la façon dont le système comprend une image. Là où les premières versions restaient piégées dans des motifs très locaux, V4 arrive enfin à saisir la forme globale du panneau et à rester fiable même quand on lui montre une image hors du dataset. C'est exactement le type de comportement qu'on attend si l'on veut un jour déployer ce modèle dans des conditions réelles.

La question du déploiement embarqué devient alors assez naturelle. Une Raspberry Pi 5 est largement capable d'exécuter un modèle comme V4 à condition de l'optimiser, par exemple en l'exportant au format ONNX. Ce type de conversion permet de réduire la latence et de rendre le modèle compatible avec différents environnements matériels, ce qui est essentiel dès qu'on veut sortir du cadre du PC de développement.

En résumé, cette progression du modèle, confrontée à des architectures « plus avancées » et à des scénarios d'usage concrets, montre qu'on peut peut-être envisager une intégration dans un système embarqué comme le turtlebot. Le travail à venir consistera surtout à affiner le pipeline, et adapter le tout aux contraintes du terrain.