Université de Strasbourg & Cerema

Rapport de stage — Master 2 CSMI

Détection automatique par IA pour caractériser le transport de matières dangereuses (TMD)

Cas du tunnel de Schirmeck

Étudiant
Dimitri KLOCKENBRING

Encadrants
Guillaume GUBLIN
Christophe HEINKELÉ

Année universitaire 2021-2022

Table des matières

Remerciements

Je souhaite adresser mes remerciements à Guillaume Gublin et Christophe Heinkelé pour avoir encadré mon stage.

Mes remerciements vont aussi à Pierre Charbonnier, Valérie Muzet et tous les membres de l'équipe ENDSUM Strasbourg pour leur accueil et les échanges que nous avons pu avoir pendant ces six mois.

Merci Colin

1 Introduction

Pour assurer la sécurité du trafic à l'intérieur des tunnels, la surveillance des véhicules transportant des matières dangereuses (TMD) est un enjeu important. Le transport de ces matières doit être signalé par des plaques spécifiques apposées à l'arrière du véhicule, et l'analyse de ces plaques permet de déterminer ce qui transite dans le tunnel. À l'heure actuelle, la collectivité européenne d'Alsace (CeA), gestionnaire du tunnel de Schirmeck (67), effectue ces comptages ponctuellement et manuellement, en observant le trafic sur le terrain ou sur bande de vidéosurveillance.

Un projet d'automatisation de ces comptage a été proposé par l'équipe de recherche d'ENDSUM Strasbourg à la CeA. Il a pour objectif de développer une solution pour le dénombrement automatique des TMD et l'identification des matières dangereuses transitant par le tunnel de Schirmeck. L'automatisation doit faciliter les comptages, les fiabiliser et permettre d'obtenir des statistiques en continu, donc plus représentatives qu'actuellement. À cette fin, deux caméras ont été installées sur site en octobre 2021. Leur système de déclenchement automatique permet d'acquérir de courtes séquences d'images à partir desquelles on souhaite détecter automatiquement les TMD et identifier leurs plaques. Pour cela, le Cerema propose de mettre en œuvre des méthodes d'apprentissage supervisé profond.

Mon stage de fin de Master s'inscrit dans le contexte de cette collaboration. J'ai travaillé au Cerema avec l'équipe de recherche d'évaluation non destructive des structures et des matériaux (ENDSUM), pour qui ce projet est innovant, car les plaques TMD sont un type de données qui n'a pas encore été étudié. Il s'agit aussi d'un défi en raison de l'important volume des données à traiter.

L'objectif principal du stage consiste à mettre en œuvre des algorithmes de reconnaissance et à évaluer leurs performances, de façon à démontrer l'utilité de l'apprentissage profond pour analyser les images de TMD. Il s'agit de choisir et d'adapter des architectures neuronales pour ce cas d'étude, de les entraîner à partir d'images annotées et de mesurer leurs performances sur des données test. Le tri et l'annotation des images sont aussi un enjeu majeur, dans la mesure où le volume des données disponibles a grandi tout au long du stage, demandant un soin particulier pour les réintégrer dans de nouveaux apprentissages.

Dans le rapport qui suit, je présenterai le contexte général du stage avant de donner quelques éléments théoriques sur l'état de l'art des méthodes d'apprentissage profond. Puis je détaillerai mes contributions pratiques. Pour cela, je présenterai dans un premier temps les méthodes que j'ai mises en œuvres pour le traitement des données et pour l'entraînement et l'évaluation des réseaux de neurones choisis, et je détaillerai dans un second temps les résultats que j'ai obtenus avec ces méthodes, avant de conclure sur leur pertinence et sur de possibles perspectives et pistes d'amélioration.

2 Contexte

2.1 Le Cerema et le groupe ENDSUM

Le centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement, ou Cerema, est un établissement public à caractère administratif créé en 2014. Placé sous la tutelle du ministère de la Transition écologique, il se compose de huit directions territoriales, dont le Cerema Est implanté à Nancy, Metz et Strasbourg, et de trois directions techniques : la DTecTV (territoire et ville), la DTecITM (infrastructures de transport et matériaux) et la DTecREM (risques, eau et mer) [1].

Les missions du Cerema s'articulent autour des thèmes de l'aménagement et du développement durable. Plus spécifiquement, ses six domaines d'action principaux sont l'environnement et les risques, la mer et le littoral, l'expertise et l'ingénierie territoriale, le bâtiment, les mobilités et les infrastructures de transport.

Dans le cadre de mon stage, j'ai travaillé à l'agence de Strasbourg, qui compte cinq groupes d'activité :

- le groupe UMR AE (unité mixte de recherche de l'acoustique et de l'environnement)
- le groupe Bâtiment, Construction, Immobilier
- le groupe Voies et Plates-Formes d'Infrastructures
- le groupe Ouvrages d'Art
- le groupe ENDSUM (évaluation non destructive des structures et des matériaux)

C'est au sein de l'équipe de recherche ENDSUM que j'ai effectué mon stage. Également présente à Angers et Rouen, une vingtaine d'agents y sont impliqués. À Strasbourg, l'équipe compte une dizaine de personnes : des chercheurs, des techniciens, un métrologue et des vacataires.

Le département ENDSUM travaille au développement et à l'évaluation de méthodes non invasives pour le diagnostic des infrastructures. Ces méthodes exploitent des données telles que des images ou des signaux électromagnétiques, et mettent notamment en œuvre des techniques de reconnaissance de formes, de reconstruction 3D (photogrammétrie), et d'intelligence artificielle. Ces développements peuvent aboutir à la création d'outils opérationnels, comme les appareils de mesure Coluroute [1] et Ecodyn [2] et les logiciels IRCAN [3] et IREVE [4].

Dans le cadre de mon stage, j'ai participé aux activités de recherche et développement du groupe ENDSUM relatives à l'analyse d'images de scènes routières, en mettant en œuvre des techniques d'intelligence artificielle (apprentissage profond). Par la suite, ces travaux pourront servir de point de départ au développement d'outils opérationnels pour la détection et l'analyse des transports de matières dangereuses.

1. Le Coluroute (COefficient de LUminance des ROUTEs) est un appareil servant à déterminer les caractéristiques photométriques des revêtements de la route, directement sur site.
2. L'Ecodyn est un réflectomètre servant à évaluer la qualité des marquages routiers
3. Imagerie Routière par Caméra Numérique
4. Imagerie Routière, Étalonnages, Visualisation, Exploitations

2.2 Règlementation du transport des matières dangereuses

Le transport de marchandises dangereuses par la route est réglementé par l'***A**ccord relatif au transport international des marchandises **D**angereuses par **R**oute*, ou **ADR** [2]. Signé en 1957 à Genève et entré en vigueur en 1968, il a été ratifié par 52 pays dans le monde [5]. En France, l'ADR est entré en vigueur le 1er janvier 1997 et a été complété en 2009 par un arrêté appelé *"Arrêté TMD"* [3].

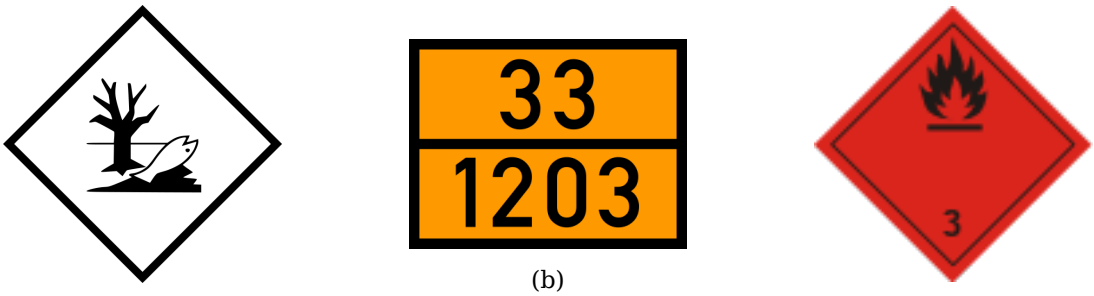
Les deux éléments principaux de l'ADR sont ses annexes A et B, régulièrement modifiées et mises à jour. L'annexe A définit quelles marchandises dangereuses sont autorisées pour le transport international, en fonction de leur nature, mais aussi de leur emballage et de leur étiquetage. L'annexe B quant à elle impose les conditions pour la construction, l'équipement, et la circulation des véhicules transportant ces marchandises. En particulier, elle impose l'apposition de plaques de signalisation à l'arrière et sur les côtés de ces véhicules. [6]. Ces plaques peuvent être de deux types :

- **Plaque étiquette**, que nous appellerons aussi *"symbole"* ou *"losange"*. Ce type de plaque en forme de losange affiche un **pictogramme** qui donne une indication générale sur le type de matière dangereuse transportée.
- **Panneau orange**, que nous appellerons aussi *"code"* ou *"rectangle"*. Ce sont des plaques de forme rectangulaire avec un fond orange. Un panneau orange peut être :
  - Un **panneau vierge**, pouvant être affiché si l'ADR n'impose pas l'ajout de plaque étiquette à l'extérieur du véhicule pour transporter la marchandise, ou si d'autres panneaux doivent être apposés ailleurs qu'à l'arrière ou à l'avant du véhicule (comme des panneaux oranges codifiés, dans le cas du transport de plusieurs types de marchandises dangereuses).
  - Un **panneau codifié** affichant un code constitué de deux parties : un **numéro d'identification du danger** [4], donnant lui aussi une information générale sur le type de la matière), et un **numéro ONU** [5] de quatre chiffres spécifiant précisément la nature de la matière.

Deux exemples de plaques étiquettes souvent observés au tunnel de Schirmeck sont visibles fig. 1 : le symbole "matières dangereuses pour l'environnement" (fig. 1a) et le symbole "liquides inflammables" (fig. 1c). Un exemple de panneau orange codifié est visible fig. 1b : le numéro d'identification du danger 33 indique un liquide hautement inflammable, et le numéro ONU 1203 indique qu'il s'agit d'essence. Enfin, la figure 2 illustre une situation où ces plaques sont apposées sur un camion citerne. D'autres exemples de symboles sont listés en annexe dans le tableau 7.

5. Au 1er janvier 2021.

6. Certaines plaques peuvent aussi être apposées à l'avant du véhicule. De plus, les dimensions des plaques sont réglementées : 30cm x 40 cm ($\pm 10\%$) pour les panneaux oranges, et un minimum de 10cm x 10cm pour les plaques étiquettes.



(a) (b) (c)

Figure 1 – Exemples de signalisation ADR : plaque étiquette NDE (nocif et dangereux pour l'environnement, fig. 1a), plaque étiquette n°3 (liquides inflammables, fig. 1c), et panneau orange 33 1203 (essence hautement inflammable, fig. 1b)

Source: Wikipedia [6]

Figure 2 – Camion citerne transportant des matières dangereuses (essence) à proximité de l'entrée Sud du tunnel de Schirmeck

2.3 Projet de comptage des transports de matières dangereuses (TMD)

Le Cerema a passé un contrat de recherche et développement en collaboration avec la Communauté européenne d'Alsace (CeA), qui opère la gestion du tunnel de Schirmeck, avec une notification du marché à la date du 05 Août 2021. Ce dernier a pour objectif *"d'automatiser le comptage des véhicules transportant des matières dangereuses via une application d'identification des TMD, ceci afin de limiter au maximum l'intervention humaine et de fiabiliser les statistiques qui pourraient être réalisées tout au long de l'année."*[7].

Dispositif de captation vidéo

Dans le cadre de ce projet, deux caméras ont été placées sur le site du tunnel de Schirmeck en octobre 2021 (fig. 3). L'une est placée à l'extérieur (tournée vers l'entrée sud du

7. Objectif énoncé dans le cachier des clauses techniques particulières du marché (CCTP)

tunnel), l'autre à l'intérieur (tournée dans la direction opposée). Ces deux caméras sont équipées d'un dispositif de déclenchement automatique qui enregistre de courtes séquences vidéo lors du passage d'un véhicule. À partir de celles-ci, on souhaite pouvoir identifier automatiquement les TMDs et décoder leurs éventuelles plaques signalétiques.

Figure 3 – Implantation des caméras à l'extérieur et à l'intérieur du tunnel de Schirmeck
Source: Google Maps

Approche par apprentissage supervisé profond Pour ce faire, la méthode proposée par le Cerema consiste à procéder par apprentissage supervisé profond. Il s'agit d'une expérience innovante pour l'équipe ENDSUM, qui bien qu'ayant déjà une expérience en imagerie et en intelligence artificielle, travaille ici pour la première fois avec ce type de caméra et ce type d'objets (plaques TMD).

Phases du projet Avant de commencer le projet, il a été décidé de le scinder en 4 phases distinctes, avec une étape de "Go/No-Go" en milieu de projet :

Phase 1 : Les images des véhicules sont disponibles sur un serveur consultable à distance

Phase 2-a : Un premier modèle est mis en place au Cerema, ses performances sur les images sont connues

Go/No-Go : À l'issue de ces deux premières phases d'expérimentation, les partenaires décident s'ils souhaitent ou non mener le projet à son terme

Phase 2-b : Le logiciel est livré. Il est opérationnel et donne les statistiques voulues

Phase 3 : Des statistiques bisannuelles sont assurées par le logiciel

Actuellement, la phase 1 n'est pas pleinement satisfaite, en raison de problèmes logistiques empêchant la consultation à distance des images vidéo. Un membre de l'équipe ENDSUM se rend donc sur place environ toutes les deux semaines pour récupérer les données depuis sur un serveur local. Malgré cette limitation, assez d'images ont pu être collectées pour que la **phase 2-a** puisse commencer, avec pour objectif de mettre en œuvre des algorithmes d'intelligence artificielle et d'évaluer leurs performances pour l'analyse des images de TMD. C'est à cet objectif que j'ai pu apporter ma contribution dans le cadre de ce stage.

2.4 Objectifs et organisation du stage

L'objectif principal de ce stage consiste dans un premier temps à entraîner des algorithmes à la reconnaissance de TMD, par des méthodes d'apprentissage profond. Dans un second temps, les performances de ces algorithmes doivent être évaluées pour mettre en évidence leur efficacité. Ce sujet est motivé par le projet du Cerema en collaboration avec la CeA pour le comptage des TMD, mais également par une volonté de l'équipe ENDSUM de se familiariser avec les outils disponibles dans le domaine de l'apprentissage profond, qui deviennent de plus en plus prometteurs pour l'analyse de tous types de signaux.

Plus spécifiquement, ce stage se découpe en plusieurs sous-objectifs :

- Pour se familiariser avec les méthodes qui seront mises en œuvre, il est important de présenter un **état de l'art** des méthodes de classification d'images et de détection d'objets utilisés en apprentissage profond
- Pour faciliter le tri (actuellement manuel) des images collectées pour le projet avec la CeA, un modèle de **classification** binaire doit être mis en œuvre et évalué
- Pour reconnaître spécifiquement les TMD, un modèle de **détection** de plaques doit être entraîné sur une base de données annotée, avant d'être lui aussi évalué. En particulier, cela nécessite de choisir une méthode d'annotation des plaques TMD et de la mettre en œuvre pour constituer des bases de données d'entraînement et de test
- Il est important d'effectuer l'**évaluation** des modèles sur des bases de test représentatives des données obtenues au cours d'une journée. En effet, les TMD étant par exemple très minoritaires parmi toutes les images disponibles, une évaluation sur une base de test avec une grande proportion de TMD aboutirait à des mesures de performances peu pertinentes
- Après avoir entraîné les modèles de détection sur des images, on souhaite aussi développer et évaluer un premier prototype d'algorithme pour le **comptage des TMD** (par véhicule)
- Enfin, tout au long du projet, un soin doit être apporté à l'**intégration continue des nouvelles images** dans le corpus de données déjà disponibles.

Outils utilisés Au cours de mon stage, j'ai principalement développé des scripts en langage Python [7], dans deux environnements Anaconda [8] (un pour la classification, un pour la détection). Les algorithmes d'apprentissage profond ont été mis en œuvre avec les librairies Keras et TensorFlow [9]. J'ai également utilisé deux programmes : *LabelImg* [10] pour l'annotation des plaques TMD, et *Object Detection Metrics* [11] pour valider mon implémentation des méthodes d'évaluation en les comparant avec un outil de référence. Tous deux sont développés en langage Python et distribués sur GitHub [12], sous licence MIT [13]. J'ai travaillé

sur une station de travail sous Windows 10, équipée à deux processeurs graphiques (GPU) `Nvidia Quadro RTX 5000` [14].

Organisation et encadrement Ce stage a été encadré par Guillaume Gublin, technicien, et Christophe Heinkelé, chercheur, tous deux travaillant au groupe ENDSUM. Des points d'avancement réguliers ont permis de suivre ma progression et de prendre des décisions sur les prochaines tâches à effectuer. Ces réunions étaient également importantes vis-à-vis du projet de comptage TMD dans son ensemble, c'est pourquoi d'autres membres du groupe pouvaient aussi y être présents : Pierre Charbonnier, chef du groupe ENDSUM et directeur du Cerema par intérim, Philippe Foucher, chercheur, et Lionel Thomas, technicien. Leur collaboration m'a aidé à de multiples niveaux : bibliographie, compréhension des modèles, tri et annotation des données (processus hautement chronophage), organisation, choix d'implémentations et des expériences à mener au cours du stage (statistiques sur les données, entraînements, évaluations...).

J'ai également pris une part active dans plusieurs réunions avec des représentants de la CeA. Pendant ces entrevues, nous avons pu présenter l'avancement de nos travaux, préciser les besoins du client et prendre des décisions sur la logistique du projet : règlementation sur la protection des données, prêt et installation de caméras, accès réseaux...

Enfin, deux séminaires se sont tenus pendant la première moitié du stage, réunissant tous les membres et stagiaires du groupe ENDSUM. Ces moments furent l'occasion de présenter les avancées de nos stages respectifs et d'échanger sur les sujets de chacun.

3 État de l'art

3.1 Généralités sur l'apprentissage profond

L'apprentissage profond (*deep learning*) fait partie du domaine de l'apprentissage machine (*machine learning*), qui désigne "l'art de programmer les ordinateurs pour qu'ils puissent apprendre à partir de données" [8]. Au coeur de l'apprentissage profond se trouvent les réseaux de neurones artificiels, ou ANN (*artificial neural networks*), qui sont des structures de nœuds interconnectés inspirées par la biologie du cerveau. Un ANN peut contenir de nombreuses couches de neurones cachées et ainsi atteindre une grande complexité. Il s'agit alors d'un réseau de neurones profond, ou DNN (*deep neural network*). Une telle architecture peut effectuer des tâches très complexes, comme de la classification d'images ou du diagnostic médical.

Le premier modèle de neurone artificiel a été proposé en 1943 dans l'article *A logical calculus of the ideas immanent in nervous activity* [16] du neurobiologiste Warren McCulloch et du mathématicien Walter Pitts. Ils proposèrent des structures simples pour effectuer des opérations logiques fondamentales (ET, OU, NON...), qui pouvaient être combinées pour représenter des propositions plus complexes. En 1957, le psychologue Frank Rosenblatt invente le *perceptron* [17], qui introduit les notions de poids des connexions entre neurones (*weights*) et de fonction d'activation (*activation function*), aujourd'hui utilisées dans tous les DNNs.

Le perceptron (1957)

Le perceptron (fig. 4) est une architecture composée d'une couche de neurones artificiels appelés TLUs (*threshold logic units*). Il reçoit plusieurs variables en entrée (*inputs*, en vert dans la figure) et renvoie plusieurs valeurs en sortie (*outputs*), une pour chaque TLU (en bleu dans la figure).

Chaque TLU est connecté à toutes les variables d'entrée, c'est pourquoi la couche qui les contient est appelée couche dense (*dense layer*, ou *fully connected layer*). À chacune de ces connexions (flèches grises dans la figure) est associé un poids. La valeur de sortie (*output*) du neurone est calculée en deux étapes :

1. Tout d'abord, une somme pondérée est calculée à partir des variables d'entrée, en prenant comme coefficients les poids associés à chaque connexion.

2. Cette somme pondérée est ensuite seuillée pour obtenir une valeur entière, typiquement 0 ou 1. Un tel seuillage est un exemple de fonction d'activation.

8. *"Machine Learning is the science (and art) of programming computers so they can learn from data"* (Aurélien Géron, [15])

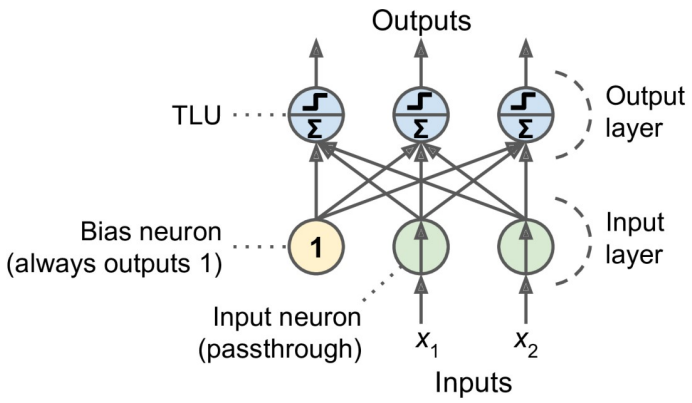


Figure 4 – Un perceptron avec deux variables d'entrée et trois variables de sortie

Source: [15]

L'architecture du perceptron se traduit dans la formule algébrique suivante :

$$h_{W,b} : X \mapsto \phi(XW + b) \tag{1}$$

Avec :

- $X = (x_1, \cdots, x_n) \in \mathbb{R}^n$ un vecteur ligne contenant n variables d'entrée, aussi appelé *échantillon* (dans la figure 4, n vaut 2).

  Il peut aussi s'agir d'une matrice de taille $b \times n$, contenant b lignes de longueur n. On dit alors que X est un *batch* (lot) de b échantillons.

- $W \in \mathcal{M}_{n,m}(\mathbb{R})$ la matrice des poids (*weights*) de la couche dense. Ici, m est le nombre de variables en sortie du perceptron (dans la figure 4, m vaut 3).

  Chacune des m colonnes est associée à un unique neurone de la couche dense, et contient les n poids utilisés pour effectuer la somme pondérée des variables d'entrée pour ce neurone. Le vecteur des m sommes pondérées obtenues est exactement XW.

  Si X est un batch de b échantillons, la sortie sera de taille $b \times m$, soit m variables de sortie pour chaque échantillon.

- En général, un vecteur $b \in \mathbb{R}^m$ est également utilisé. Il contient des poids qui vont s'ajouter à chaque somme pondérée. Ce vecteur s'appelle le *biais*.

- Enfin, ϕ est la fonction d'activation. Pour des TLUs, une fonction souvent utilisée est la fonction de Heaviside (graphe fig. 6) :

$$\phi(s) = \begin{cases} 0 \text{ si } s < 0 \\ 1 \text{ si } s \geq 0 \end{cases} \tag{2}$$

  La fonction d'activation ϕ est appliquée terme à terme à chaque composante du vecteur $XW + b$.

Perceptron multicouche (MLP)

La structure linéaire du perceptron original, à une seule couche, ne permet pas de résoudre certains types de problèmes (comme le problème du "OU Exclusif"). Cependant, beau-

coup de ces limitations sont résolues en enchaînant plusieurs perceptrons les uns à la suite des autres. Le modèle ainsi obtenu (fig. 5) est appelé perceptron multicouche, ou MLP (*multilayer perceptron*). Il est composé d'une couche d'entrée (*input layer*), d'une ou plusieurs couches denses cachées (*hidden layers*) et d'une couche de sortie (*output layer*).[9] Tous les neurones d'un MLP sont interconnectés : cette architecture est donc aussi appelée réseau entièrement connecté (*fully connected network*).

Chaque couche après la couche d'entrée est un perceptron (régi par l'équation 1) prenant comme entrée la sortie de la couche précédente. Ainsi, la donnée se propage en sens unique, du début à la fin du réseau : on parle de réseau à propagation directe (*feedforward*).

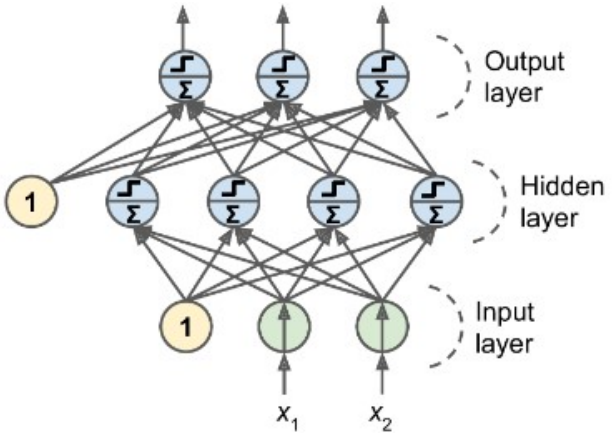


Figure 5 – Perceptron multicouche à deux entrées, une couche cachée de quatre neurones et trois sorties

Source: [15]

D'un point de vue algébrique, l'équation 1 modifiée pour correspondre au MLP de la figure 5 devient :

$$h : X \mapsto h_{W_2,b_2} \circ h_{W_1,b_1}(X) = \phi\left((\phi(XW_1 + b_1))W_2 + b_2\right) \tag{3}$$

Où $W_1 \in \mathcal{M}_{2,4}(\mathbb{R}),\ b_1 \in \mathbb{R}^4$ sont les poids et le biais de la couche cachée de quatre neurones, et $W_2 \in \mathcal{M}_{4,3}(\mathbb{R}),\ b_2 \in \mathbb{R}^3$ sont les poids et le biais de la couche de sortie à trois neurones. Les fonctions d'activation sont ici importantes, car sans elles le modèle serait toujours linéaire, de poids W_1W_2 et de biais $(b_1W_2+b_2)$. L'ajout de la couche cachée n'apporterait donc aucune complexité :

$$h : X \mapsto (XW_1 + b_1)W_2 + b_2 = X(W_1W_2) + (b_1W_2 + b_2) \tag{4}$$

Le perceptron multicouche est une des architectures les plus utilisées aujourd'hui dans le domaine de l'apprentissage profond.[10] Dans le cadre de stage, le MLP a été l'un des composants de modèles de classification d'image que j'ai mis en œuvre.

9. Un MLP avec un grand nombre de couches cachées est en particulier un réseau neuronal profond, ou DNN. La notion de "grand nombre de couches" est floue et varie selon les époques et les personnes.

10. Les fonctions d'activation des MLPs actuels ne sont plus les mêmes : de fonctions discrètes (comme le seuillage du perceptron), on est passé à des activations continues. Ce changement a été nécessaire pour entraîner les modèles par rétropropagation de gradient (section 3.1)

Usage des réseaux de neurones

Les réseaux de neurones peuvent être utilisés pour résoudre divers problèmes d'apprentissage sur des données. Ceux-ci peuvent être répartis en plusieurs grandes catégories [11] :

- **Apprentissage supervisé** (*supervised learning*) : c'est le type de problème le plus courant. Dans ce contexte, les données d'entraînements sont annotées, c'est-à-dire qu'elles incluent les solutions attendues, appelées vérités terrain (*ground truth*). Les problèmes de régression et de classification sont des exemples typiques d'apprentissage supervisé.
  La **régression** consiste à prédire une variable réelle à partir d'une donnée d'entrée, par exemple le prix d'un appartement en fonction de sa surface.
  La **classification** consiste à prédire une classe à partir d'une donnée d'entrée, par exemple, décider si une image représente un chat ou un chien. Typiquement, cette prédiction est encodée dans un vecteur dont chaque composante, comprise entre 0 et 1, est associée à une classe. Si le modèle prédit l'appartenance à une classe avec un haut niveau de confiance, la composante correspondante aura une valeur proche de 1.

- **Apprentissage non supervisé** (*unsupervised learning*) : dans ce type de problème, les données d'entrées ne sont pas annotées, le modèle apprend de lui-même à en extraire des propriétés intéressantes. Deux exemples de problèmes non supervisés sont le **partitionnement de données** (*clustering*), qui consiste à partager un ensemble en sous-ensembles de données partageant des caractéristiques communes, et la **réduction de dimension** (*dimensionality reduction*), qui vise à construire une représentation en petite dimension d'une donnée initiale en grande dimension, en perdant le moins d'information possible. Ceci est très utile pour pouvoir effectuer des traitements qui prendraient beaucoup trop de temps en grande dimension.

- **Apprentissage semi-supervisé** (*semisupervised learning*) : dans ce type de problème, une partie des données est annotée et l'autre non. Par exemple, un programme de reconnaissance des visages dans un album photo est capable de regrouper automatiquement les images d'une même personne (apprentissage non supervisé), mais a besoin que le nom de la personne soit fourni sur une de ces images afin d'identifier la personne sur les autres images (apprentissage supervisé).

- **Apprentissage par renforcement** (*reinforcement learning*) : il consiste à rechercher une stratégie optimale pour résoudre un problème. Dans ce type de problème, un *agent* évolue dans un environnement dans lequel il peut effectuer diverses actions. Selon le contexte, une action rapporte une récompense ou une pénalité plus ou moins importante, qui doit permettre à l'agent d'apprendre une stratégie optimale. Un exemple célèbre est l'algorithme AlphaGo [18], développé par Google en 2015, qui a été le premier programme à battre un joueur professionnel au traditionnel jeu de go.

Dans le cadre de ce stage, les modèles de classification d'images et de détection d'objets mis en œuvre s'inscrivent dans le contexte de l'apprentissage supervisé. [12]

11. Ces problèmes font partie de l'apprentissage machine en général, les réseaux de neurones ne sont donc pas la seule façon de les aborder.

12. La détection est à la fois un problème de régression et de classification

Entraînement par rétropropagation de gradient

Malgré la diversité des problèmes existants, l'approche adoptée pour entraîner un réseau de neurones est souvent la même : l'objectif à atteindre est traduit en une *fonction de coût* (*loss function*) que l'on cherche à minimiser. De nombreuses fonctions de coût existent en fonction du problème abordé. Dans le contexte de l'apprentissage supervisé, les fonctions de coût expriment une distance entre les prédictions du modèle et les réponses attendues (vérité terrain). [13]

Une fois qu'une *loss* a été définie, on cherche les poids du modèle qui vont la minimiser. Ceux-ci dépendent des données d'entraînement et des vérités terrain associées. Il est important de noter que dans ce contexte, ces données sont des constantes : les seules variables à ajuster sont les poids du modèle. L'algorithme fondamental utilisé pour effectuer cette optimisation est la **descente de gradient**. C'est un procédé itératif qui met à jour les poids du modèle avec la règle suivante :

$$\theta \leftarrow \theta - \eta \nabla_\theta(Loss(\theta)) \tag{5}$$

Où

- θ est la variable contenant tous les poids du modèle
- $Loss$ est la fonction de coût que l'on souhaite minimiser
- $\nabla_\theta(Loss(\theta))$ est le gradient de la fonction de coût par rapport aux poids du modèle.
- η est un hyperparamètre à choisir appelé taux d'apprentissage (*learning rate*) : avec un taux élevé, l'entraînement converge plus rapidement, mais devient aussi plus sensible aux variations (bruit) dans les données, ce qui peut restreindre les capacités de généralisation du modèle.

L'optimisation spécifique des réseaux de neurones *profonds* a longtemps été un défi, leur complexité rendant difficile le calcul du gradient de la fonction de coût. Mais en 1985, la méthode de *rétropropagation du gradient* (*backpropagation* [19]) fut popularisée et devint la plus communément utilisée pour l'entraînement des DNNs. [14] Cet algorithme rend possible un calcul efficace du gradient. Il repose sur la règle de la chaîne (*chain rule*) [21], un résultat fondamental du calcul différentiel qui permet d'exprimer le gradient d'une composée de fonctions comme composée des gradients de chaque fonction prise individuellement. Schématiquement, une étape d'optimisation par rétropropagation fonctionne ainsi :

1. La donnée d'entrée est passée à travers le réseau (*forward pass*), et tous les résultats des couches intermédiaires sont gardés en mémoire.
2. La fonction de coût est appliquée pour calculer l'erreur.
3. La règle de la chaîne est utilisée pour calculer la contribution de la sortie du modèle à l'erreur.
4. Le modèle est parcouru en sens inverse (*backward pass*), en utilisant la règle de la chaîne à chaque couche pour mesurer sa contribution à l'erreur. On obtient ainsi le gradient de la *loss* en fonction de tous les poids du modèle.

13. Les fonctions de coût utilisées en régression et en classification seront explicitées section 3.3 et section 3.4
14. L'idée avait déjà été proposée auparavant, notamment par Paul Werbos en 1974 [20]

5. Finalement, une descente de gradient est effectuée pour mettre à jour tous les poids du réseau.

Il est important de noter que, pour que la descente de gradient soit applicable, le gradient doit pouvoir être défini. Ceci nécessite des propriétés de dérivabilité de la fonction de coût utilisée, mais aussi des fonctions d'activation à l'intérieur du modèle. En particulier, la fonction de seuillage utilisée dans le perceptron et dans les premiers MLPs n'a pas les bonnes propriétés, puisque qu'elle est discrète, donc de gradient nul presque partout.[15] C'est pourquoi la publication [19] de 1985 n'a pas seulement proposé la rétropropagation comme méthode pour le calcul de gradient, mais a aussi modifié l'architecture des perceptrons multicouches en utilisant une nouvelle fonction d'activation, la **sigmoïde**, dont la dérivée ne s'annule pas :

$$\sigma(s) = \frac{1}{1+e^{-s}} \tag{6}$$

Aujourd'hui, à l'intérieur des réseaux de neurones, la sigmoïde laisse souvent place à une fonction plus appelée **ReLU** (*rectified linear unit*) eq. (7). Celle-ci est plus simple et a l'avantage d'être plus rapide à calculer. Cependant, la sigmoïde est encore utilisée, notamment à la sortie des réseaux pour les problèmes de classification. La fonction ReLU n'est pas dérivable en 0, mais en pratique cela n'empêche pas d'utiliser la descente de gradient pour l'entraînement.

$$ReLU(s) = \max(0, s) \tag{7}$$

Les graphes de la fonction de Heaviside eq. (2), de la sigmoïde eq. (6) et de la fonction ReLu eq. (7) sont comparés dans la figure 6.

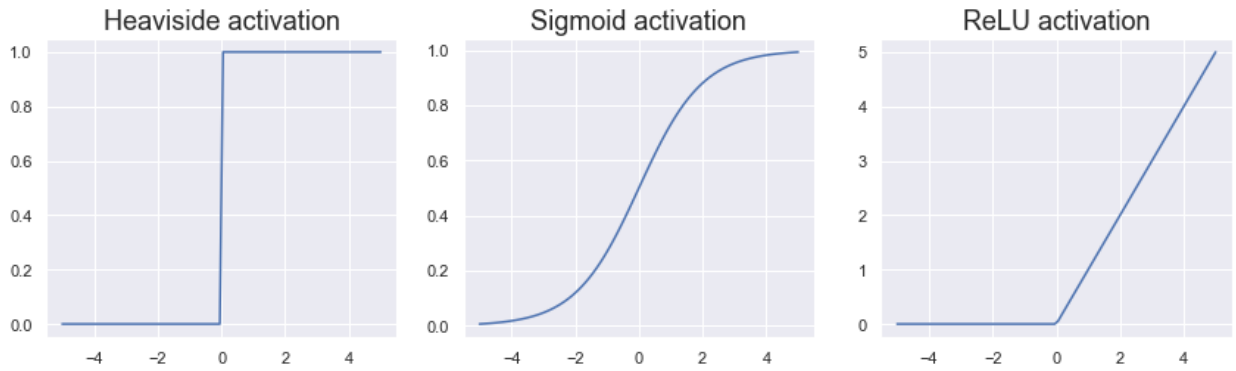


Figure 6 – Comparaison de trois fonctions d'activations : la fonction de Heaviside, discrète, a une dérivée nulle presque partout, ce qui n'est pas le cas de la sigmoïde et de la fonction ReLU

Dans le cadre de ce stage, les modèles mis en œuvre ont été entraînés avec des optimiseurs calculant le gradient de la *loss* par rétropropagation. Concrètement, ces algorithmes sont des variations de la descente de gradient. L'un des plus célèbres, que nous avons utilisé, est l'algorithme **Adam** (*adaptive moment estimation*), proposé pour la première fois en 2014 [23].

15. Le perceptron n'était donc pas entraîné par descente de gradient, mais avec une autre méthode inspirée de la *règle de Hebb* [22]

3.2 Réseaux neuronaux convolutifs (CNN)

Le sujet de ce stage portant sur de l'analyse d'image, l'architecture fondamentale utilisée dans ce domaine est le **réseau de neurones convolutif**, ou **CNN** (*convolutional neural network*). Ces structures inspirées de la vision naturelle permettent d'extraire des **caractéristiques** utiles des images (*feature maps*), qui peuvent ensuite être facilement analysées par des modèles plus classiques, comme des perceptrons multicouches [16].

Le grand nombre de variables dans une image rend en effet difficile l'utilisation directe de réseaux entièrement connectés comme le MLP pour traiter des images, car leur connecter une couche dense nécessite de définir un très grand nombre de poids. [17] En comparaison, les CNNs sont des architectures partiellement connectées (chaque neurone de sortie n'est *pas* connecté à tous les neurones d'entrée), et dont les poids sont partagés (chaque neurone de sortie a les mêmes poids que tous les autres), ce qui limite fortement le nombre de poids à entraîner dans chaque couche.

Convolution en deux dimensions

Les CNNs reposent sur l'opération de convolution, en deux dimensions dans le contexte de l'analyse d'images. Une convolution en 2D prend une image en entrée et renvoie une image en sortie. À chaque pixel de l'image d'entrée où la convolution est appliquée, on associe une somme pondérée des valeurs des pixels situés dans un voisinage proche de celui-ci. Les poids de cette somme sont définis dans le **noyau** (*kernel*) de la convolution, une petite matrice qui parcourt chaque position dans l'image d'entrée avec un pas de déplacement à choisir (*stride*) [18]. Ainsi, chaque pixel de l'image finale est entièrement défini par une zone spécifique de l'image d'origine : c'est le **champ réceptif** du pixel (*receptive field*).

16. Les CNNs sont aussi très utiles dans d'autres domaines, comme la reconnaissance de voix et le traitement automatique des langues, ou NLP (*natural language processing*)

17. Par exemple, une image de 100 pixels de côté à trois canaux de couleurs contient $3 \times 100^2 = 30000$ variables. Si on souhaite la connecter à une première couche dense de 1000 neurones, la matrice des poids doit être de taille 30000×1000, soit déjà trente millions de poids à optimiser.

18. Pour simplifier, la *stride* est fixée à 1 pixel dans les explications qui suivent

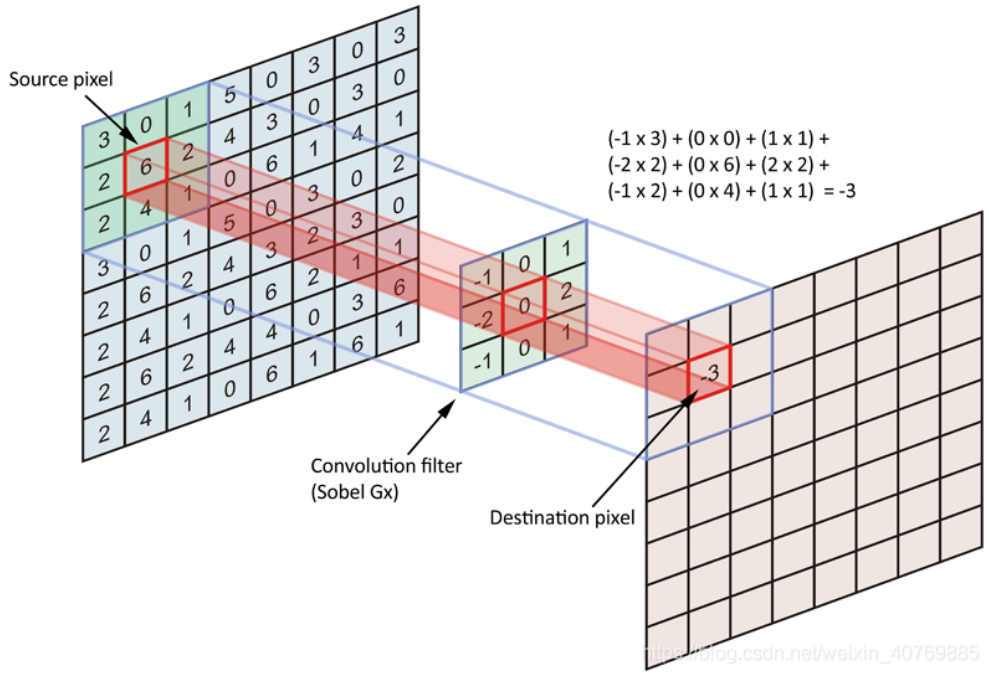


Figure 7 – Fonctionnement schématique de la convolution 2D

Source: [24]

Le schéma de fonctionnement de la convolution est représenté fig. 7. Ici, le noyau de convolution est composé d'un **filtre** de taille 3×3 et le pixel de destination encadré en rouge est la somme des neufs pixels correspondants dans l'image source, pondérée par les poids du noyau.

Pour des images d'entrée et de sortie en noir et blanc, c'est-à-dire à un seul canal, la convolution se traduit par la formule suivante [19] :

$$Conv(X, W)_{i,\,j} = \sum_{-k \leq di, dj \leq k} X_{i+di,\,j+dj} \times W_{k+di,\,k+dj} \tag{8}$$

Où

- $X \in \mathcal{M}_{w,h}(\mathbb{R})$ est l'image d'entrée, de largeur w et de hauteur h.
- $W \in \mathcal{M}_{2k+1}(\mathbb{R})$ est le noyau de la convolution, de taille $(2k+1) \times (2k+1)$, avec k est un petit nombre entier, généralement 1 ou 2 (dans la figure 7, k vaut 1, le noyau est de taille 3). Le noyau est donc de taille impaire, ce qui lui permet d'avoir un centre bien défini.
- $Conv(X, W)_{i,\,j}$ est le coefficient de coordonnées (i, j) dans l'image résultant de la convolution. La taille de l'image de sortie (et donc le domaine de définition de i et j) peut varier. En effet, le noyau de la convolution ne peut pas recouvrir sans dépasser les pixels situés à une distance du bord strictement inférieure à k. Les deux approches les plus courantes pour résoudre ce problème sont les suivantes :
  - On peut étendre le domaine de définition de l'image en définissant des valeurs pour les pixels qui n'existent pas. Typiquement, on leur assignera la valeur 0, mais on peut

19. Toutes les indexations se font en partant de 0.

aussi prolonger l'image avec des pixels semblables à ceux du bord de l'image, par exemple avec des procédés de symétrie. En procédant ainsi, l'image de sortie aura la même taille que l'image d'entrée. Cette méthode est appelée *padding* (remplissage)

- Quand aucun *padding* n'est utilisé, on ignore tout simplement les pixels du bord, ce qui résulte en une image de sortie plus petite, de taille $(w-2k)\times(h-2k)$. On dit que la convolution est effectuée en mode valide.

En pratique, l'image d'entrée aura souvent un nombre variable de canaux d'entrées (nb_channels_in), typiquement trois (rouge, vert, bleu). De même, l'image de sortie aura un nombre arbitraire de canaux (nb_channels_out), aussi appelés **filtres**. Par ailleurs, on peut effectuer la convolution sur un batch de plusieurs images. La formule 8 peut être modifiée pour donner la valeur du pixel (i,j) dans le canal c_{out} de la $b^{\text{ième}}$ image du batch. Dans ce cas, le noyau de la convolution n'est plus une matrice, mais un tenseur de taille $(2k+1)\times(2k+1)\times$ nb_channels_in $\times$ nb_channels_out :

$$Conv(X,W)_{b,\ i,\ j,\ c_{out}} = \sum_{\substack{-k\leq di,dj\leq k \\ 0\leq c_{in}<\text{nb\_channels\_in}}} X_{b,\ i+di,\ j+dj,\ c_{in}} \times W_{k+di,\ k+dj,\ c_{in},\ c_{out}} \tag{9}$$

Exemples de convolution sur une image

La convolution met en avant les caractéristiques (*features*) de l'image qui sont similaires aux caractéristiques de son noyau. La figure 8 présente deux exemples sur une image d'entrée (fig. 8a) en noir et blanc, de taille 380×380 (image extraite des données collectées au tunnel de Schirmeck dans le cadre de ce stage).

1. Le premier noyau utilisé est de taille 31 ($k=15$). Il s'agit d'une matrice de zéros, à l'exception de la ligne centrale dont tous les coefficients valent 1. La convolution a été effectuée en mode valide, l'image de sortie (fig. 8b) est donc de taille $380-2k=330$. On observe que les lignes horizontales présentes sur l'image de départ sont encore visibles, alors que les lignes très différentes (comme les lignes verticales de l'échelle) ont tendance à s'estomper.

2. Le deuxième noyau est un noyau de Sobel [25], de taille 3 ($k=1$) :

$$W = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

   Il a une structure similaire au premier[20] , mais au lieu d'avoir une seule ligne horizontale, il en a deux qui sont complémentaires. L'idée est de mettre en évidence par une simple différence les variations horizontales dans l'image. Là encore, la convolution a été faite en mode valide, l'image de sortie (fig. 8c) est donc de taille $380-2k=378$. Cette fois-ci, les lignes horizontales ressortent encore plus et les bords verticaux ont quasiment disparu.

20. Les 2 n'ont pas tellement d'importance et auraient pu être remplacés par des 1

(a) Image d'entrée (380×380)

(b) Exemple de convolution (330×330) avec un noyau de taille 31

(c) Autre exemple (378×378) avec un noyau de taille 3

Figure 8 – Deux exemples de convolution sur une image originale de taille 380x380 (fig. 8a)

Architecture générale d'un CNN

Il existe de nombreux modèles de réseaux de neurones convolutifs, mais leurs architectures reposent le plus souvent sur les mêmes éléments fondamentaux :

- **Couches convolutives** : c'est l'élément central de tout CNN, dont le fonctionnement est décrit par l'équation 9. Une couche convolutive est définie par quatre hyperparamètres principaux qui ont été décrits plus haut : son nombre de canaux de sortie, ou **filtres** (noté nb_channels_in dans eq. (9)), la taille de son noyau, (***kernel size***), les pas de déplacement du noyau dans les directions horizontale et verticale (***strides***), et son mode de ***padding***.
  En général, les couches convolutives sont aussi suivies d'une fonction d'activation ReLU

- **Couches de pooling** : le *pooling* est un procédé de sous-échantillonnage utile pour réduire la complexité du réseau. Plusieurs algorithmes de *pooling* existent. Un exemple très simple est le *maxpooling* : une fenêtre glissante parcourt l'image et sélectionne à chaque endroit la valeur de pixel la plus grande. On obtient ainsi une image plus petite (typiquement, la résolution sera divisée par deux, ce qui divise par quatre le nombre de pixels).

- **Couches entièrement connectées** : en général, les couches convolutives ne font qu'extraire des caractéristiques (*feature maps*) de l'image d'entrée. Une fois que ces *feature maps* sont extraites, elles sont en général repassées à travers un réseau dense (section 3.1) pour obtenir une prédiction en sortie du modèle. [21]

Un CNN classique (fig. 9) est constitué d'une alternance de couches convolutives et de couches de pooling : le nombre de filtres de couches convolutives augmente au fur et à mesure, mais la résolution de l'image diminue grâce aux couches de pooling. En sortie du réseau, la partie dense renvoie la prédiction finale.

21. Le passage à la partie dense nécessite une réorganisation des variables : toutes les valeurs des *feature maps* sont alignées dans un vecteur unidimensionnel

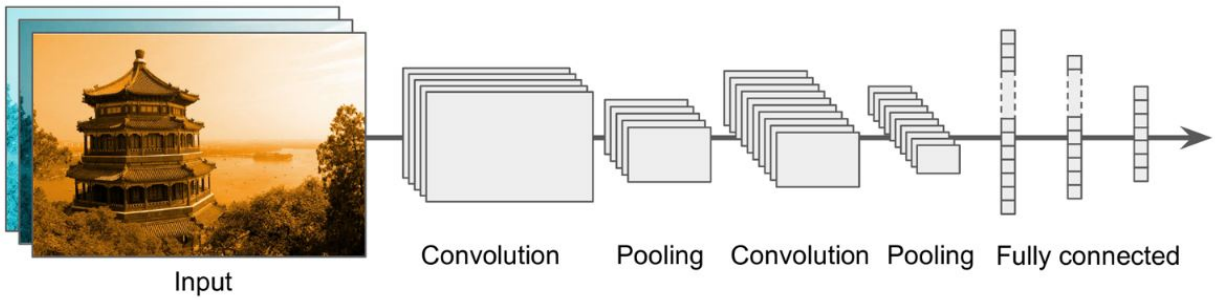


Figure 9 – Architecture caractéristique d'un réseau de neurones convolutif (CNN)

Source: [15]

3.3 Classification d'images

3.3.1 Principe de fonctionnement et exemple d'architecture (VGG16)

La classification est l'un des problèmes les plus courants dans le domaine de l'apprentissage profond. Il s'agit d'associer une ou plusieurs classes à une variable d'entrée.

Le plus souvent, la classification est réalisée par un perceptron multicouches (fig. 4) connectant toutes les variables d'entrée à un vecteur de sortie de longueur égale au nombre de classes possibles. Chaque coefficient de ce vecteur aura une valeur entre 0 et 1. La prédiction peut être multi-classes, auquel cas une fonction d'activation *sigmoïde* (eq. (6)) est généralement utilisée avant la couche de sortie. Dans le cas d'une prédiction à une seule classe, la sortie doit être un vecteur de probabilités (dont la somme des coefficients vaut 1). On utilise alors généralement une autre fonction d'activation appelée *softmax* [26].

L'entraînement d'un modèle de classification se fait avec une fonction de coût appelée **crossentropie**. Si y est un vecteur de vérité terrain (à valeurs dans $\{0, 1\}$) et p est un vecteur de prédictions (à valeurs dans $[0, 1]$), la distance crossentropique $C(p, y)$ est donnée par l'équation 10. Il s'agit d'une fonction non bornée, dont le graphe est tracé fig. 10.

$$C(p, y) = -\sum_i y_i \times \ln(p_i) \tag{10}$$

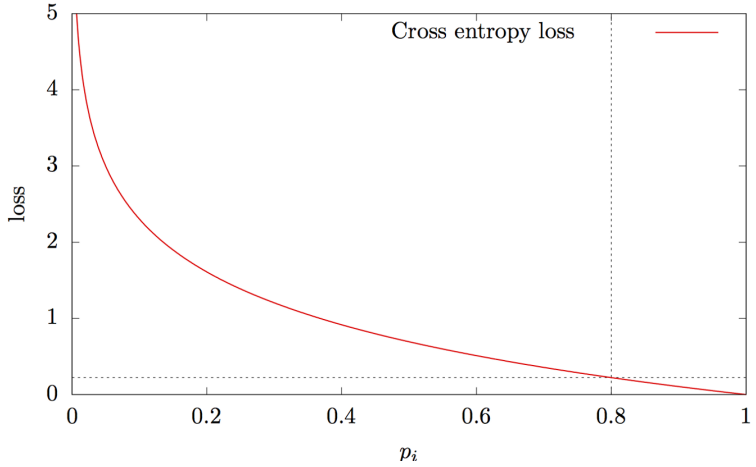


Figure 10 – Graphe de la *loss* de crossentropie (eq. (10))

Source: [27]

Pour la classification d'images spécifiquement, le MLP est rarement utilisé seul. C'est possible pour de toutes petites images, comme les chiffres manuscrits du jeu de données MNIST. Mais pour d'autres applications, le MLP est la plupart du temps couplé à un réseau CNN, qui est capable d'extraire les caractéristiques importantes de l'image d'entrée (*feature maps*) tout en réduisant progressivement sa taille grâce aux couches de *pooling* alternées avec les couches convolutives.

De nombreuses architectures pour la classification d'image ont été développées au fil des années. Certaines des plus célèbres sont AlexNet (2012, [28]), Inception (2014, [29]), R-CNN (2014, [30]) et ses évolutions Fast R-CNN [31] et Faster R-CNN [32], VGG16 (2015, [33]), ResNet (2016, [34]) et MobileNet (2017, [35]).

Architecture VGG16 Un exemple représentatif de modèle de classification d'images est le VGG16 [33], développé en 2015. Ce modèle prend en entrée une image à trois canaux de taille 224×224, et prédit en sortie une classe parmi 1000 possibilités. Il se nomme ainsi car il comporte seize couches de neurones (treize couches convolutives et trois couches denses). Son architecture (fig. 11) consiste en une série de convolutions successives alternées avec des couches de pooling, qui réduisent progressivement la dimension de l'image d'entrée tout en augmentant le nombre de canaux. La fin du modèle est un réseau dense qui effectue la classification avec une activation *softmax* (une seule classe prédite).

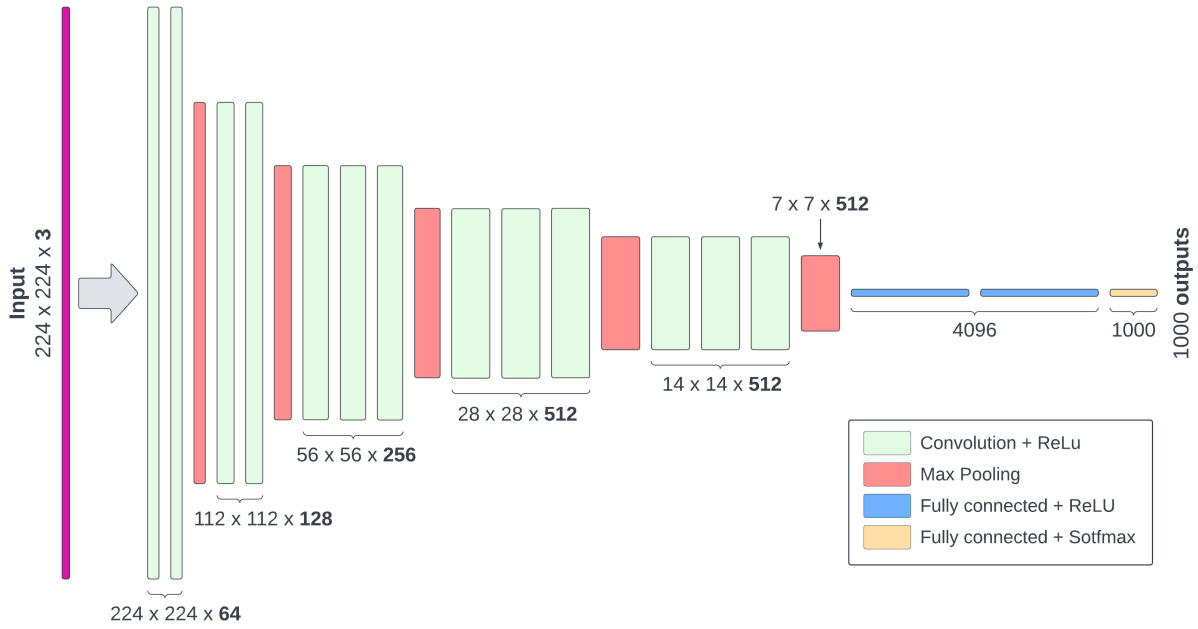


Figure 11 – Architecture du VGG16 ([33])

3.3.2 Évaluation d'un modèle de classification

L'évaluation d'un modèle de classification d'images repose fondamentalement sur la comparaison des prédictions du modèle avec les réponses attendues, qui constituent ce qu'on appelle la vérité terrain (*ground truth).* Les mesures de performances sont effectuées classe par classe. Quatre indicateurs fondamentaux sont utilisés pour les définir :

- **Vrais positifs (TP)** : nombre de prédictions positives dont la vérité terrain est effectivement positive

- **Faux positifs (FP)** : nombre de prédictions positives pour lesquelles la vérité terrain est en réalité négative.
- **Vrais négatifs (TN)** : nombre de prédictions négatives dont la vérité terrain est effectivement négative
- **Faux négatifs (FN)** : nombre de prédictions négatives pour lesquelles la vérité terrain est en réalité positive

Ces indicateurs peuvent être résumés dans une matrice de confusion, dont un exemple est visible fig. 12. Dans cette matrice, il y a trois classes A, B, C. Chaque ligne (en bleu) correspond à une classe de la vérité terrain et chaque colonne (en rouge) correspond à une classe prédite. La somme des coefficients d'une ligne donne le nombre d'images de la vérité terrain appartenant à la classe associée. La somme des coefficients d'une colonne donne le nombre d'images prédites dans la classe associées. On lit ainsi facilement les indicateurs. Par exemple, la classe B a ici 2 vrais positifs, 2 faux positifs et 1 faux négatif. Pour les vrais négatifs, il faut sommer toutes les quantités en dehors de la ligne B et de la colonne B. Ici, cela donne 5 vrais négatifs.

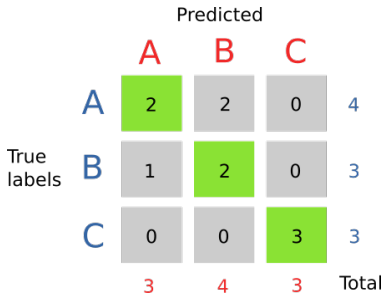


Figure 12 – Exemple d'une matrice de confusion avec trois classes

Source: [36]

Les TP, FP, TN, FN sont ensuite utilisés pour calculer plusieurs métriques, dont les deux plus importantes sont

- Le **rappel** par classe (eq. (11)) : cette mesure évalue la capacité du modèle à trouver les échantillons de la classe en calculant la proportion de vrais positifs parmi toutes les vérités terrain positives (le rappel ne dépend que de l'effectif de la classe).

$$\text{Rappel} = \frac{TP}{TP + FN} = \frac{TP}{\text{vérités positives}} \tag{11}$$

- La **précision** par classe (eq. (12)) : cette mesure évalue la fiabilité des prédictions du modèle en calculant la proportion de vrais positifs parmi toutes les prédictions (la précision dépend aussi des effectifs des autres classes, et est donc très sensible à la distribution des classes).

$$\text{Précision} = \frac{TP}{TP + FP} = \frac{TP}{\text{prédictions positives}} \tag{12}$$

Deux autres mesures plus générales peuvent être définies :

- Le **score F1** par classe (eq. (13)), qui est la moyenne harmonique de la précision et du rappel

$$\text{F1} = \frac{1}{\frac{1}{\text{Rappel}} + \frac{1}{\text{Précision}}} = \frac{\text{Rappel} \times \text{Précision}}{\text{Rappel} + \text{Précision}} \tag{13}$$

- L'**accuracy** par classe, qui est la proportion de toutes les prédictions correctes parmi tous les échantillons

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} = \frac{\text{prédictions correctes}}{\text{tous les échantillons}} \tag{14}$$

Pour l'évaluation des modèles mis en œuvre dans ce stage, les principales mesures qui nous intéresseront sont le rappel et la précision.

Enfin, un dernier critère à prendre en compte est le **seuil de décision** du modèle. En effet, la prédiction d'un classifieur pour une classe est un nombre entre 0 et 1. Le choix (arbitraire) d'un seuil permet de définir la frontière de décision entre une prédiction positive et une prédiction négative : on considèrera que la prédiction est positive si le seuil de décision est dépassé, et négative sinon.

Le choix du seuil de décision a donc une grande influence sur les TP, FP, TN, FN, et par extension sur le rappel et la précision : un seuil de décision faible privilégie un rappel élevé au détriment de la précision, et réciproquement, un seuil de décision élevé privilégie une précision élevée au détriment du rappel. En pratique, on tracera donc les **courbes de rappel, précision et score F1** en fonction du seuil de décision pour avoir une idée plus précise des performances du modèle. Enfin, le compromis entre précision et rappel peut être observé isolément en traçant la courbe de la précision en fonction du rappel.

3.4 Détection d'objets

3.4.1 Principe de fonctionnement et exemple d'architecture (RetinaNet)

La détection d'objets est un problème plus complexe que la classification. Elle consiste à la fois à localiser un objet dans une image et à le classifier. Concrètement, la localisation s'effectue à l'aide d'un rectangle appelé **boîte englobante** (*bounding box*) placé autour de l'objet détecté. Chaque boîte englobante est labellisée avec la classe de l'objet qu'elle contient.

Pour générer leurs prédictions, les modèles de détection utilisent souvent des **ancres**. Il s'agit d'un ensemble de boîtes englobantes prédéfinies (position, échelle, ratio) recouvrant de façon homogène la surface de l'image (fig. 13). Chaque ancre est susceptible de détecter un objet dont la position et les dimensions sont similaires aux siennes.

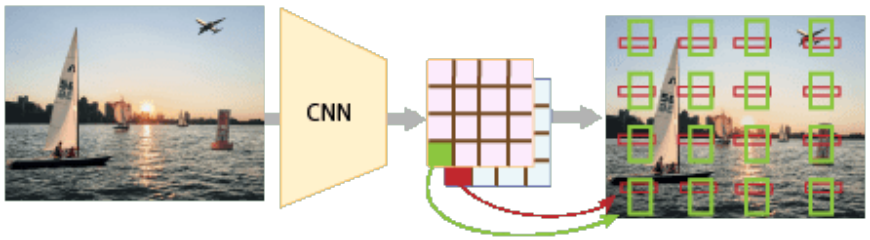


Figure 13 – Disposition initiale des ancres dans un détecteur d'objets

Source: [37]

En pratique, une ancre est représentée par un vecteur de 4 valeurs (nécessaires pour décrire sa position et sa forme) et subit deux opérations :

- Une opération de **régression** qui prédit un ajustement (*offset*) à appliquer aux coordonnées de son vecteur
- Une opération de **classification** prédisant la nature de son contenu

Selon la nature exacte de cette classification, un détecteur peut être de type *two-shot* (détection en deux étapes) ou *single-shot* (détection en une seule étape) :

- Dans un détecteur *two-shot*, cette classification des ancres vise uniquement à décider si leur contenu est un objet ou non, mais sans préciser la classe exacte de l'objet. Il s'agit donc d'une classification binaire ("arrière-plan" contre "objet"). Après cette étape, les ancres contenant effectivement un objet doivent être classifiées une seconde fois (classification multiclasse) pour déterminer la nature exacte des objets détectés.
- Dans un détecteur *single-shot*, une seule classification (multiclasse) a lieu : une ancre peut se voir attribuer la classe "arrière-plan", mais aussi n'importe quelle autre classe d'objet.

Les détecteurs *two-shot* sont plus lents à l'inférence, car ils nécessitent de classifier des ancres deux fois d'affilée. Mais en contrepartie, ils peuvent souvent atteindre une meilleure précision. Le Faster-RCNN ([32], 2015) (Faster regions with CNN features) est un exemple notable de cette catégorie de détecteurs.

Les détecteurs de type *single-shot* peuvent être plus difficile à entraîner, mais sont beaucoup plus rapides à l'inférence, car leurs ancres n'ont besoin d'être classifiées qu'une seule fois. Le SSD (Single Shot Multibox Detector [38], 2016), le détecteur YOLO (You Only Look Once [39], 2015) et le RetinaNet ([40], 2018) sont des exemples importants de détecteurs *single-shot*. Plus récemment, le détecteur FCOS (Fully Convolutional One-Stage Object Detection [41], 2019) a également fait son apparition.

Architecture RetinaNet Un exemple de détecteur d'objets est le RetinaNet [40], modèle de type *single-shot* proposé en 2018. Son architecture est résumée sur la figure 14 : ce modèle utilise un ResNet [34] pour l'extraction des caractéristiques des images, et recombine ces informations dans un *Feature Pyramid Network* (FPN [42]), qui est une méthode proposée en 2017 pour obtenir une hiérarchie de *feature maps* fortement sémantique à toutes les échelles. De plus, il utilise deux sous-réseaux entièrement *convolutifs* pour la régression et la classification des ancres.

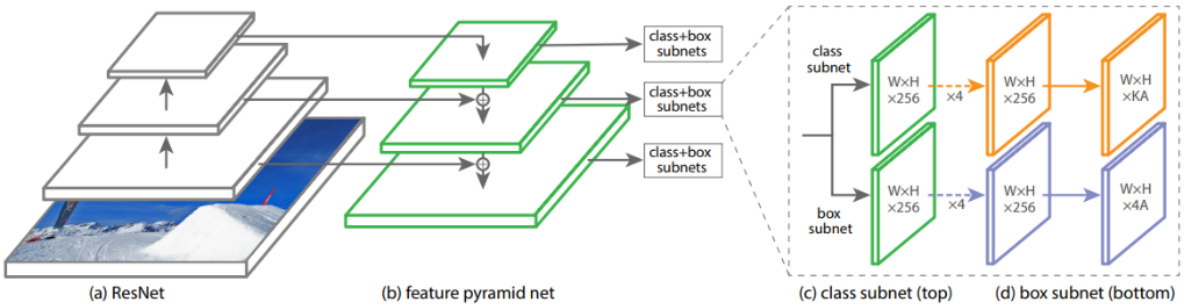


Figure 14 – Architecture du RetinaNet [40]

Source: [40]

Mais l'innovation majeure du RetinaNet se trouve dans l'utilisation d'une nouvelle fonction de coût pour l'entraînement appelée *Focal Loss*, qui permet d'atténuer un problème de surreprésentation de l'arrière-plan commun en détection.

En effet, une raison avancée dans [40] pour expliquer la difficulté à entraîner les détecteurs *single-shot* est la contribution massive des exemples "faciles" (arrière-plan) dans la *loss* de classification. La figure 15 illustre ce problème. Sur la courbe de crossentropie habituelle (en rouge), on peut voir qu'une bonne prédiction avec un score de confiance de 0.8 contribue tout de même sensiblement à la *loss*. Ainsi, même si le détecteur apprend rapidement à bien classer les exemples d'arrière-plan, leur surreprésentation a tendance à surcharger la loss, ce qui ne permet pas d'apprendre efficacement les autres classes d'objets (exemples positifs minoritaires). La *Focal Loss* permettet d'atténuer significativement ce problème par l'introduction d'un facteur multiplicatif à la formule de la crossentropie (en bleu dans l'équation 15).[22]. La courbe de cette nouvelle fonction est tracée en bleu sur la figure fig. 15 : on y observe que les prédictions correctes avec un haut score de confiance ne contribuent quasiment plus à la loss globale.

$$FL(p, y) = -\sum_i y_i \times (1 - p_i)^\gamma \times \ln(p_i) \quad \gamma = 2 \tag{15}$$

22. Ce facteur est toujours entre 0 et 1. Il est plus proche de 1 à quand p_i tend vers 0, et plus proche de 0 quand p_i tend vers 1. Il en résulte que la valeur de crossentropie originale sera d'autant plus réduite que le score de confiance est grand. Avec un exposant $\gamma = 0$, on retrouve la crossentropie originale.

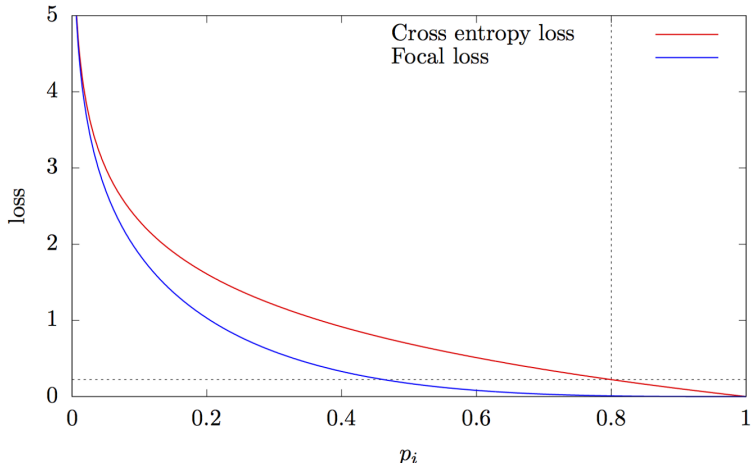


Figure 15 – Graphe de la Focal Loss [40]

Source: [27]

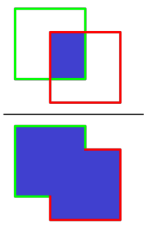
L'efficacité de la *Focal Loss* pour l'entraînement sur des jeux de données déséquilibrés a fortement contribué au succès du RetinaNet (cette *loss* est maintenant implémentée dans la librairie Tensorflow).

3.4.2 Évaluation d'un modèle de détection

L'évaluation des modèles de détection d'objets est plus complexe qu'en classification. Par exemple, il n'y a plus de correspondance directe entre les objets de la vérité terrain et les objets prédits : il faut donc redéfinir les notions de vrais et faux positifs ou négatifs. Une fois ceci fait, il faut également définir des métriques d'évaluation. À l'heure actuelle, il n'y a pas de consensus à ce sujet : chaque jeu de données (COCO, Pascal VOC...) implémente des mesures légèrement différentes, et les outils proposés pour l'évaluation utilisent des formats de fichiers d'annotation qui leur sont propres (Pascal VOC XML, COCO JSON, YOLO TXT...), ce qui a tendance à ralentir l'unification des différentes approches. Dans [43], les auteurs comparent certaines des métriques les plus couramment utilisées, qui reposent sur les mêmes principes mais présentent quelques différences algorithmiques (tracer de courbe, interpolation, intégration). Cette section a pour objectif de présenter la démarche dans ses grandes lignes.

Intersection Over Union (IOU) pour valider une détection L'évaluation d'un détecteur d'objets se fait pour chaque classe individuellement. Avant de compter les vrais positifs (TP), faux positifs (FP) et faux négatifs (FN), on doit définir ce qu'est une détection valide. Pour cela on utilise la notion d'*(Intersection over Union)* (IOU), qui est le rapport d'aires entre l'intersection et l'union de deux boîtes englobantes :

$$\text{IOU} = \frac{\text{area of overlap}}{\text{area of union}} = \frac{}{}$$



Source: [43]

Soient $B_{\text{gt}} = (x_{1_{\min}}, y_{1_{\min}}, x_{1_{\max}}, y_{1_{\max}})$ et $B_{\text{pred}} = (x_{2_{\min}}, y_{2_{\min}}, x_{2_{\max}}, y_{2_{\max}})$ deux boîtes englobantes, l'une provenant de la vérité terrain et l'autre proposée par le détecteur, exprimons les longueurs des côtés de l'intersection de ces deux boîtes :

- $\Delta_x = \max(0, \quad \min(x_{1_{\max}}, x_{2_{\max}}) - \max(x_{1_{\min}}, x_{2_{\min}}))$
- $\Delta_y = \max(0, \quad \min(y_{1_{\max}}, y_{2_{\max}}) - \max(y_{1_{\min}}, y_{2_{\min}}))$

L'IOU de ces deux boîtes peut alors s'obtenir ainsi :

$$\text{IOU}(B_{\text{gt}}, B_{\text{pred}}) = \frac{\Delta_x \cdot \Delta_y}{\text{Aire}(B_{\text{gt}}) + \text{Aire}(B_{\text{pred}}) - \Delta_x \cdot \Delta_y}$$

En se fixant un seuil τ_{iou}, typiquement égal à 0.5 (ou supérieur), il devient possible de définir une détection valide : B_{pred} détecte B_{gt} si $\text{IOU}(B_{\text{gt}}, B_{\text{pred}}) \geq \tau_{\text{iou}}$.

Comptage des vrai positifs, faux positifs et faux négatifs Toujours en considérant uniquement les boîtes d'une classe donnée, on peut maintenant choisir si une boîte compte comme un vrai positif (TP), faux positif (FP), ou faux négatif (FN) :

- Dans fig. 16a et fig. 16b, l'IOU est trop basse ou nulle : la boîte verte (vérité terrain) compte comme un FN et la boîte rouge (prédiction) comme un FP.
- Dans fig. 16c, la boîte rouge ne détecte aucune boîte de la vérité terrain et compte donc comme un FP.
- Dans fig. 16d, l'IOU est supérieure à 0.5, la détection est valide : les deux boîtes comptent comme un unique TP.
- Dans fig. 16e, la boîte verte n'est détectée par aucune boîte prédite et compte donc comme un FN.
- Dans fig. 16f, il y a deux détections valides : la boîte verte et la boîte rouge avec l'IOU la plus grande comptent comme un unique TP, l'autre boîte rouge (détection redondante) compte comme un FP.

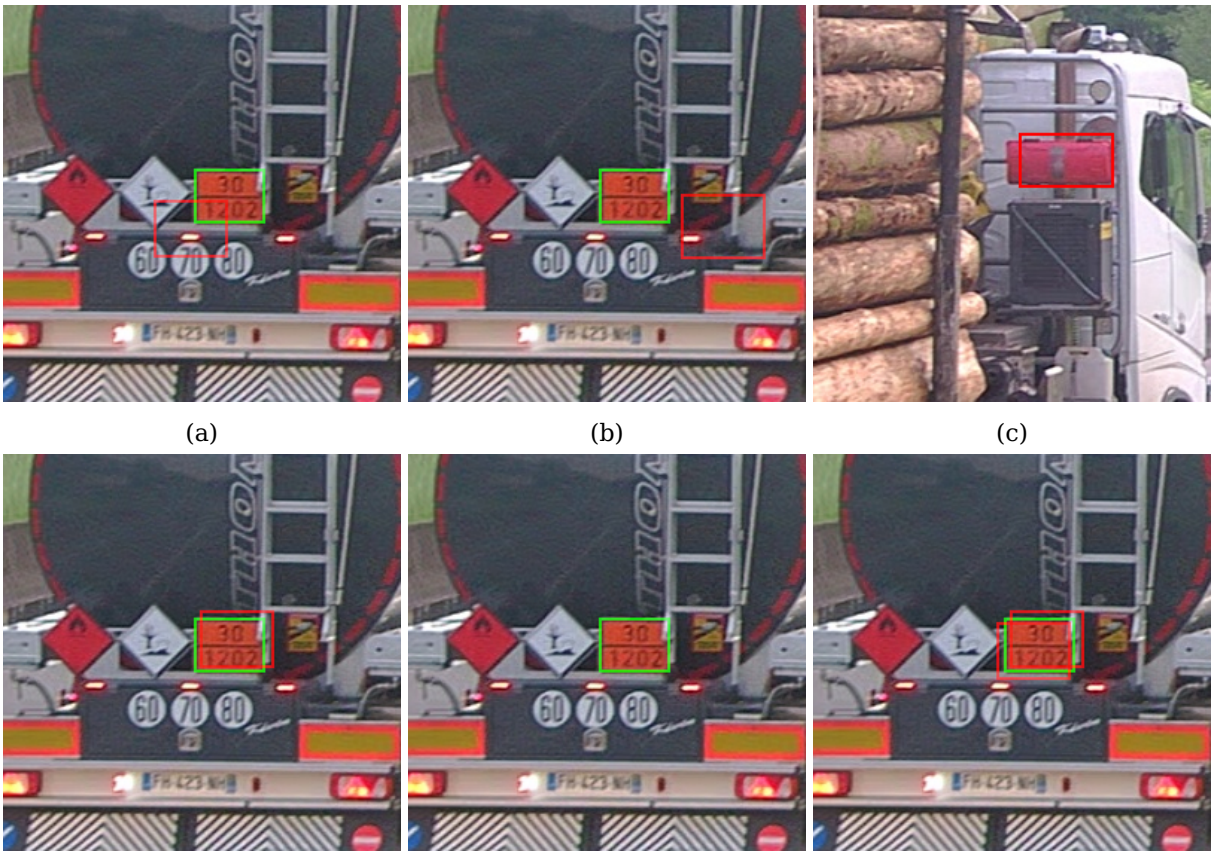


Figure 16 – Cas possibles lors d'une détection (vérité terrain en vert, prédictions en rouge)

Au final, les nombres de vrais positifs (TP), faux positifs (FP) et faux négatifs (FN) peuvent se définir ainsi :

- FN est le nombre de boîtes de la vérité terrain pour lesquelles il n'y a aucune détection valide (c.-à-d. détection avec une IOU supérieure au seuil fixé).
  Ici, il y a trois faux négatifs (fig. 16a, fig. 16b, fig. 16e)
- TP est le nombre de boîtes de la vérité terrain pour lesquelles il y a au moins une détection valide. C'est aussi le nombre de boîtes prédites qui sont des détections valides non redondantes (c.-à-d. détection avec la plus grande IOU parmi toutes les détections valides).
  Ici, il y a deux vrais positifs (fig. 16d, fig. 16f)
- FP est le nombre de boîtes prédites qui ne sont pas des détections valides ou qui sont des détections valides redondantes.
  Ici, il y a quatre faux positifs (fig. 16a, fig. 16b, fig. 16c, fig. 16f)

Dans ce contexte, on ne s'intéresse pas aux vrais négatifs (TN) qui font partie de l'arrière-plan.

Calcul des précisions moyennes (AP, mAP) Une fois que les TP, FP, FN sont définis par le choix d'un seuil d'IOU (et d'un seuil de confiance), les courbes de rappel, précision et score F1 peuvent être tracées en faisant varier le seuil de décision, comme pour l'évaluation d'un classifieur. Dans le cadre de la détection d'objets, il est cependant très courant de tra-

cer également la courbe de la précision en fonction du rappel, et de l'intégrer pour obtenir une métrique unique résumant les performances du modèle. Pour une classe, cette intégrale est appelée précision moyenne (**AP**, ou *Average Precision*). On définit aussi la **mAP** (*mean Average Precision*) comme la moyenne de toutes les AP.

4 Mise en oeuvre

4.1 Acquisition et prétraitement des données

L'acquisition et le traitement des données obtenus au tunnel de Schirmeck ont demandé un travail important tout au long de ce stage. Dans cette section, je présenterai les grandes étapes de ce flux de travail : enregistrement des vidéos sur site, détramage et tri des images, et enfin annotation des plaques TMD en vue d'entraîner un modèle de détection par apprentissage supervisé profond.

Mon degré d'implication n'a pas été le même dans toutes ces étapes, qui ont avant tout été un travail d'équipe : la captation des vidéos a été mise en place avant mon arrivée, je n'ai donc pas été sollicité pour cette première partie. De même, le détramage des vidéos et le tri des images ont été faits en grande partie par d'autres membres de l'équipe ENDSUM. J'y ai contribué modérément en détramant et en triant certaines des nouvelles données. En revanche, j'ai été fortement impliqué dans l'annotation des plaques TMD (choix du programme, lecture et gestion des formats d'annotation, annotation des images utilisées pour l'entraînement). Enfin, à la fin du stage, j'ai été beaucoup aidé pour annoter de grandes quantités d'images en vue de leur utilisation dans une base de test pour l'évaluation des modèles.

4.1.1 Enregistrement des séquences vidéo

Le dispositif de captation installé au tunnel de Schirmeck (section 2.3) comporte deux caméras qui disposent chacune d'un système de déclenchement automatique :

- Une caméra XNO-6120R-FNP [44] placée à l'extérieur, qui enregistre les poids lourds entrant par le côté sud du tunnel. Actuellement, ces vidéos sont enregistrées en résolution moyenne 800x600, à 6 im/s. D'autres résolutions ont été expérimentée (le détail des différentes configurations est explicité en annexe section 8).
- Une caméra AXIS Q1615-E Mk II [45] placée à l'intérieur, qui enregistre les poids lourds entrant par le côté nord du tunnel. Actuellement, ces vidéos sont enregistrées en haute résolution 1920x1080, à 50 im/s.

Chacune est accompagnée d'un programme fourni par le constructeur qui permet de configurer les réglages habituels (résolution, fréquence d'images...). Mais ces programmes sont aussi nécessaires pour définir les règles pour le **déclenchement automatique** : l'utilisateur peut choisir une zone dans l'image où des vidéos seront enregistrées si la caméra y détecte un mouvement. La figure 17 présente l'interface graphique pour configurer la zone de déclenchement de la caméra extérieure :

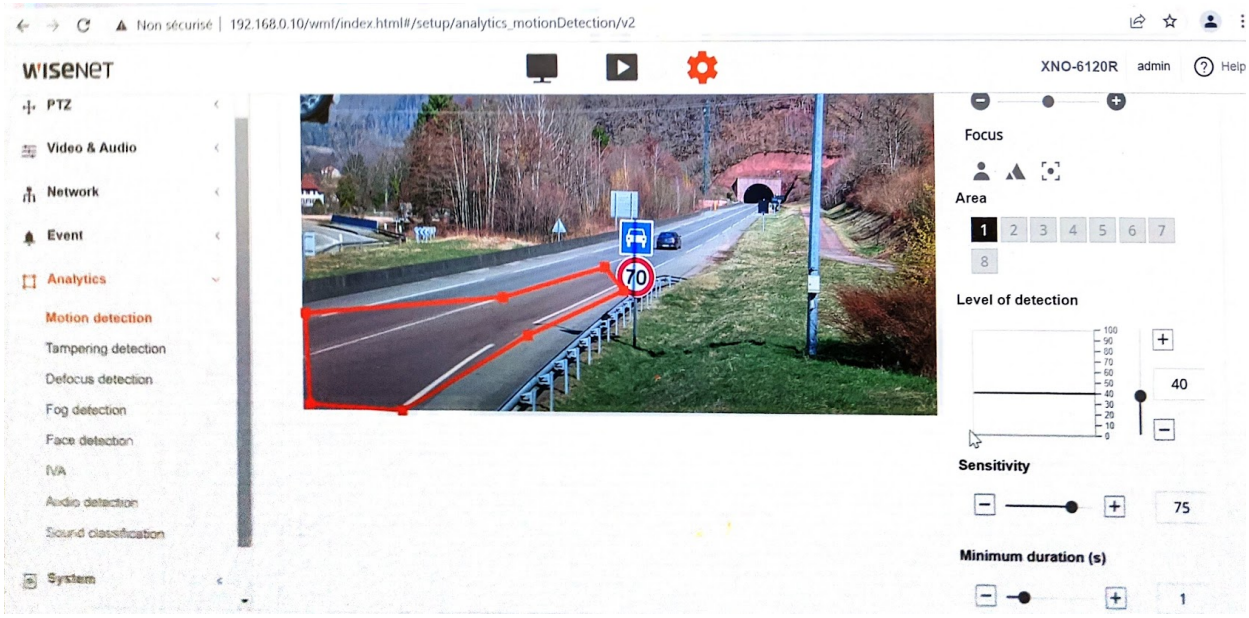


Figure 17 – Interface graphique de la caméra extérieure (XNO-6120R-FNP)

Les vidéos issues des caméras sont stockées sur un serveur NAS situé dans un local à proximité de l'entrée du tunnel. Toutes les deux semaines environ, un agent de l'équipe ENDSUM se rend sur site pour récupérer les dernières vidéos et éventuellement procéder à des réglages sur les caméras.

Actuellement, ce système présente certaines limites. Tout d'abord, les images obtenues à l'intérieur du tunnel (fig. 18) sont d'une qualité insuffisante, en raison du flou de bougé provoqué par la faible luminosité. Au cours des réunions avec l'équipe, la possibilité de "déflouter" les images par des méthodes de déconvolution a été évoquée. Cependant, dans le cadre de mon stage, il a été décidé d'exploiter uniquement les images de la caméra extérieure. De plus, nous enregistrons également des vidéos de nuit, mais les séquences obtenues (fig. 19) pour l'instant sont trop floues et les déclenchements ne se font pas au bon moment, ce qui ne permet pas d'enregistrer les passages de poids lourds correctement.

La limitation majeure reste cependant le système de déclenchement automatique des caméras, indispensable pour obtenir des comptages de TMD fiables. Cette fonctionnalité s'est avérée difficile à configurer, et nous avons observé des périodes avec des enregistrements trop peu fréquents (peu de séquences) ou déclenchés au mauvais moment (séquences sans passage de véhicule). Il a cependant été possible de récolter assez de données pour entraîner les modèles de classification et de détection. De plus, les visites sur site régulières ont permis peu à peu d'affiner le paramétrage des caméras et donc d'améliorer le déclenchement. Il est important de préciser que pour cette raison, différentes résolutions et cadrages sont présents dans nos jeux de données, en fonction de la période d'enregistrement des vidéos (configurations détaillées en annexe, section 8). Lors du traitement des données, il a donc fallu être vigilant pour assurer la cohérence des images les unes avec les autres. Cela a nécessité un redimensionnement des images en fonction de leur résolution, pour que les objets aient partout les mêmes proportions.

Figure 18 – Exemple d'images mettant en évidence le flou de bougé à l'intérieur du tunnel

Figure 19 – Exemple d'image de nuit non exploitée

4.1.2 Détramage

La première étape du traitement des données consiste en un détramage des fichiers vidéo, afin de les convertir en séries d'images plus faciles à manipuler pour les annotations et les entraînements.

Ce détramage s'est fait en utilisant la version Python de la librairie OpenCV [46]. Pour limiter la quantité de données générées, une *stride* ("foulée") est définie pour enregistrer les images à une certaine fréquence. Par exemple, avec une *stride* égale à 6, seule une image sur 6 est enregistrée. Le choix de ce paramètre peut varier, car depuis la mise en place des caméras, plusieurs fréquences d'images ont été utilisées à différentes périodes (voir annexe section 8). Par ailleurs, le cadrage de la caméra et la résolution de l'image ont également changé au long du projet. L'ensemble de ces variations sont répertoriées en annexe (section 8).

Les images issues du détramage sont classées dans des dossiers par mois et par jour. Chacune se voit attribuer un nom unique contenant le jour et l'heure de l'enregistrement, le numéro de séquence vidéo (chaque jour, cette numérotation est réinitialisée), et le numéro de l'image extraite dans la séquence. Par exemple, l'image `154_20220517_053847_6.jpg` est la 6e image extraite de la séquence n°154 de la journée du 17 mai 2022, enregistrée à 05h 38min 47s. Enfin, un fichier **journal** au format JSON est généré en sortie du détramage. Ce fichier

indique pour chaque vidéo détramée le nombre d'images extraites et la correspondance entre chaque image et sa position exacte dans la vidéo d'origine, ce qui peut être utile quand on souhaite refaire un détramage en s'assurant de la cohérence des données. Un court exemple d'un tel fichier journal est donné en annexe (section 8).

Un exemple de détramage est donné figure 20. Cette série de 43 images a été obtenue à partir du fichier vidéo `152_20220401_132028.avi` enregistré le premier avril 2022. La vidéo d'origine dure 13 secondes à 20 im/s, et une image sur 6 a été enregistrée :

Figure 20 – Images issues du détramage d'une vidéo

4.1.3 Tri des images

Pour préparer le terrain au travail d'annotation nécessaire avant l'entraînement d'un détecteur, les images issues du détramage sont triées en cinq classes disjointes :

- **TMD** : images de véhicules avec des plaques signalant le transport de matière dangereuse
- **Bus** : images de bus
- **Grumiers** : images de véhicules transportant des troncs d'arbre
- **Autres camions** : images de poids lourds qui ne sont pas dans une des catégories précédentes
- **V.Inexploitable** : (pour "véhicules inexploitables") toutes les autres images, incluant des véhicules légers et des images de la route sans circulation.

La figure 21 illustre chacune de ces classes par une courte séquence d'images

Figure 21 – Images triées en cinq classes

Avant mon arrivée, le tri des images était entièrement manuel. Cela représente un travail considérable (plus de 10 000 images à trier par jour en moyenne), qui a démarré avant mon arrivée sur le projet. Lorsque mon stage a commencé, j'ai mis en œuvre un modèle de classification binaire pour automatiser une partie du tri, en classant d'emblée les images inexploitables (mise en œuvre détaillée section 4.2.1). En effet, la figure 22 montre le poids massif de cette classe qui représente 87% des images triées. Une fois entraîné, ce modèle a pu être incorporé dès l'étape du détramage pour effectuer un premier tri "V.Inexploitable contre le reste du monde", ce qui a permis d'alléger considérablement la quantité de travail à fournir pour terminer le tri. [23]

23. Cependant, cette approche présente certaines limites : au vu de la très grande quantité d'images inexploitables attendues, même avec un modèle d'une bonne précision, les images de poids lourds désignées comme inexploitables (faux positifs) pourront représenter une part non négligeable des images de poids lourds recherchées.

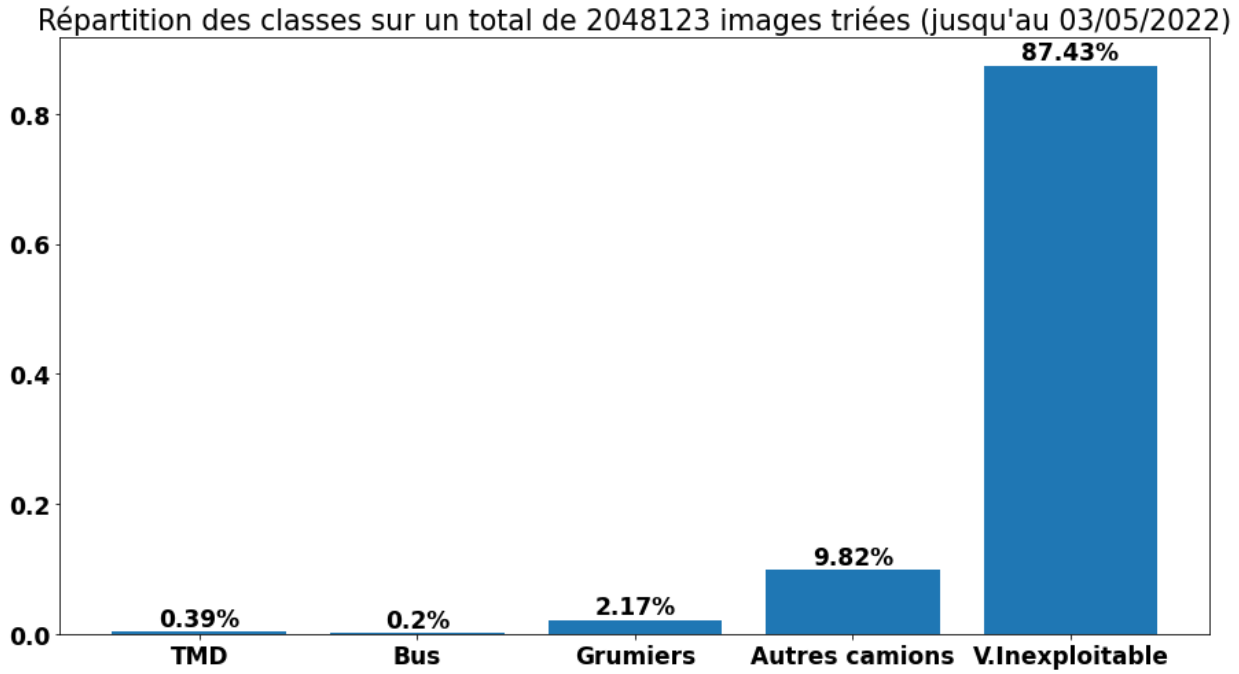


Figure 22 – Répartition des images triées en cinq classes

4.1.4 Annotation des plaques TMD

Afin de préparer l'entraînement d'un modèle de détection de plaques TMD, une annotation des images est nécessaire. Il s'agit de tracer un rectangle autour de chaque objet d'intérêt, appelé boîte englobante (ou *bounding box*). Chaque boîte englobante doit aussi être labellisée avec une étiquette décrivant son contenu.

Lors de mon stage, seules les images de la classe TMD ont été annotées, mais il est possible que des images d'autres catégories soient utilisées dans la suite du projet. De plus, il m'a été possible de prélever des images dans ces catégories pour enrichir les entraînements avec des exemples négatifs.

Pour l'annotation, nous avons utilisé LabelImg, un programme libre distribué sous licence MIT [13], dont le code source et les instructions d'utilisation sont accessibles sur un dépôt GitHub [10]. La figure 23 présente l'interface du programme ainsi qu'un exemple d'annotation de plaques TMD.

Figure 23 – Interface du programme LabelImg

LabelImg propose deux formats d'annotation : le format PascalVOC, encodé dans des fichiers XML (par défaut) et le format YOLO, encodé dans des fichiers TXT. Plus d'informations sur ces deux formats sont données en annexe, **??**. Pour ces annotations, le format PascalVOC par défaut a été retenu.

La politique d'annotation a évolué au long du processus. Au départ, un maximum de plaques ont été annotées : les plaques à l'arrière des véhicules, sur le côté, mais aussi les supports de plaques vides (un support rectangulaire est visible fig. 23, non annoté). Cette politique a ensuite été simplifiée, à présent seules les plaques visibles à l'arrière du véhicule sont annotées, et les supports de plaques ou plaques sur le côté des véhicules sont ignorés. Plusieurs raisons ont guidé ce choix :

- L'annotation est ainsi plus rapide
- On ne souhaite pas que le détecteur encadre les supports de plaques vides. Ne pas les annoter doit permettre, lors de l'entraînement, de faire comprendre au modèle que ces objets sont à considérer comme de l'arrière-plan.
- Les plaques sur le côté sont plus difficiles à détecter et à déchiffrer. De plus, il s'agit soit d'une information redondante dans le cas des losanges (les losanges sur le côté sont les mêmes qu'à l'arrière), soit d'une information illisible dans le cas des plaques rectangulaires (exemple fig. 24). La CeA nous a confirmé que seule la détection des plaques à l'arrière du véhicule est requise, et qu'une deuxième caméra filmant le côté des véhicules serait nécessaire pour obtenir le reste des informations.

Figure 24 – Codes illisibles sur le côté d'un TMD

La liste détaillée des labels utilisés pour l'annotation est fournie en annexe, table 7. Les labels ont deux niveaux d'information :

- Tout d'abord, on indique si la plaque est une plaque étiquette (en forme de losange) ou une plaque affichant un code danger (forme de rectangle). Pour cela, on utilise respectivement les mots-clés `symb` et `code`.
- Puis, on précise la nature exacte de la plaque. La plupart des pictogrammes des plaques étiquette ont un numéro qui leur est propre, les autres sont désignés par un mot. Pour les plaques danger, le code complet est recopié.

Par exemple, le label `adr_symb_8` est utilisé pour désigner une plaque étiquette signalant une matière corrosive, et le label `adr_code_33_1203` désigne une plaque de code danger signalant un carburant essence hautement inflammable. Le préfixe `adr` est utilisé pour toutes les plaques, dans l'éventualité que d'autres types d'objets soient un jour aussi annotés (par exemple des plaques d'immatriculation).

4.2 Classification de véhicules

Pour accélérer le tri des images, j'ai implémenté une méthode de classification automatique avec un modèle de classification binaire séparant les images en deux classes. Pour ce faire, deux approches étaient possibles :

- Une classification "Inexploitable contre le reste du monde", notée **noPL/PL** (non poids lourd / poids lourd), où **noPL** désigne les images inexploitables et **PL** les autres images (TMD, Bus, Grumiers, Autres camions)[24]
  Il s'agit de l'objectif principal, qui a pour but de faciliter le tri des données.
- Une classification "TMD contre le reste du monde", notée **TMD/noTMD**, où **TMD** désigne les images de TMD et **noTMD** toutes les autres images (Bus, Grumiers, Autres camions, Inexploitables). Il s'agit d'un objectif plus expérimental, préliminaire à l'entraînement d'un véritable détecteur de plaques TMD

Elles ont été mises en œuvres toutes les deux. Pour ce faire, un réseau VGG16 a été implémenté, entrainé puis évalué.

24. Cette appellation est un abus de langage, car un TMD n'est pas forcément un poids lourd

4.2.1 Implémentation d'un réseau VGG16

Pour entraîner ces classifieurs binaires, le choix a été fait d'utiliser une architecture VGG16 (voir section 3.3.1), certes rudimentaire, mais rapide à mettre en œuvre. L'implémentation s'est faite avec les librairies Keras (gpu-2.6.0) et Tensorflow (gpu-2.6.0) dans un environnement Anaconda dédié.

Nous sommes partis de l'architecture VGG16 originale, qui est déjà fournie dans la librairie Tensorflow avec des poids préentraînés. Quatre type d'objets composent sa structure : la classe `Conv2D` (pour les couches convolutives), la classe `MaxPool2D` (pour le sous-échantillonnage entre chaque série de convolutions), la classe `Dense` pour la partie entièrement connectée, et la classe `Flatten` qui est utilisée pour convertir une image en un vecteur de nombres unidimensionnel. Contrairement aux classes `Conv2D` et `Dense`, les classes `MaxPool2D` et `Flatten` n'ont aucun poids à entraîner.

Le VGG16 original ayant été conçu pour faire de la classification à 1000 classes, la fin du réseau a dû être adaptée pour correspondre à un problème de classification binaire :

- La couche dense de sortie (à 1000 neurones) a été retirée
- Après les deux couches denses (à 4096 neurones) restant à la fin du modèle, une nouvelle couche dense à 1024 neurones (avec une activation ReLU eq. (7)) a été ajoutée.
  En comptant le biais, cette couche contient $(4096 + 1) \times 1024 = 4\,195\,328$ paramètres.
- Enfin, une couche de sortie à un seul neurone (avec une activation sigmoïde eq. (6)) a été placée en sortie du réseau.
  En comptant le biais, cette couche contient $(1024 + 1) \times 1 = 1025$ paramètres.

L'architecture obtenue comporte 138 456 897 paramètres en tout. Elle est schématisée fig. 25, avec les couches ajoutées encadrées en pointillés.

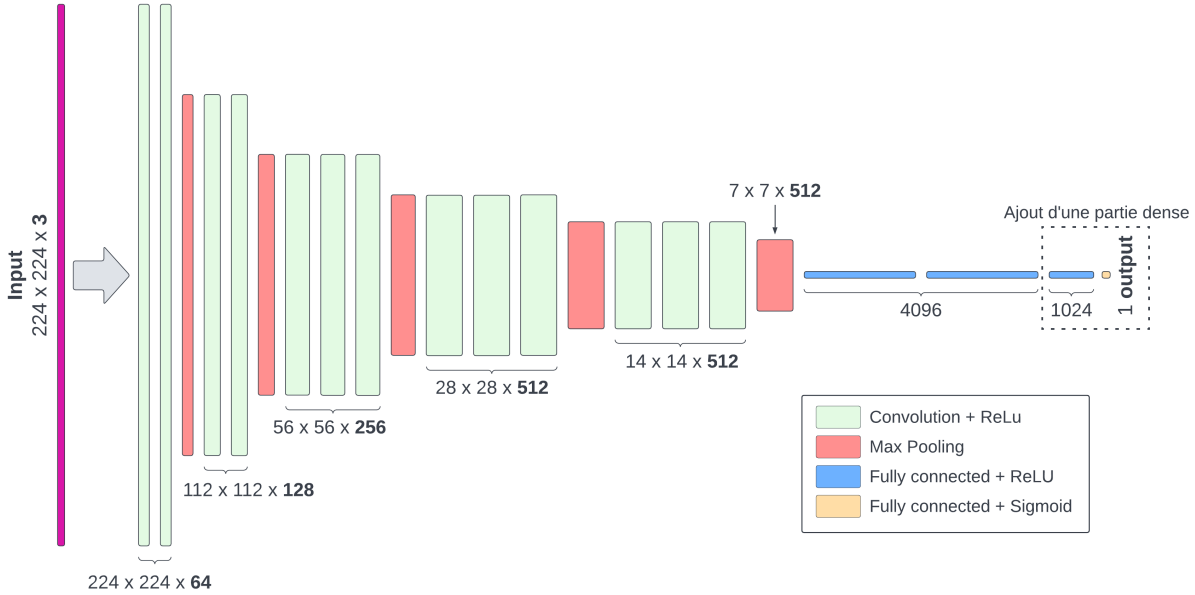


Figure 25 – Architecture du VGG16 [33] modifié pour les besoins d'une classification binaire

4.2.2 Bases de données d'entraînement et de test

Pour entraîner et évaluer les deux modèles de classification binaire (**noPL/PL** et **TMD/-noTMD**), un total de trois bases de données ont été préparées :

- Une base d'entraînement *d'apprentissage* pour le classifieur **noPL/PL** et sa base de test *d'apprentissage* associée (90% entraînement, 10% test). Pendant les entraînements, la base d'apprentissage a elle-même été divisée en une partie pour l'entraînement (85%) et une partie pour la validation (15%)
- Une base d'entraînement *d'apprentissage* pour le classifieur **TMD/noTMD** et sa base de test *d'apprentissage* associée (90% entraînement, 10% test). Pendant les entraînements, la base d'apprentissage a elle-même été divisée en une partie pour l'entraînement (85%) et une partie pour la validation (15%)
- Une base de test *applicative* pour tester les deux modèles. Cette base est constituée de la totalité des images de 10 journées triées manuellement. Elle est importante pour mesurer l'efficacité des modèles en conditions "réelles", par opposition aux conditions d'entraînement où les distributions des données peuvent ne pas être représentatives de la réalité.

Les images utilisées proviennent toutes de journées d'enregistrement situées entre le 20 octobre 2021 et le 14 février 2022. À noter que les 10 journées utilisées pour la base de test *applicative* sont aussi les journées dont sont extraites les images des bases de test *d'apprentissage*. Ces 10 journées ont été réservées à l'utilisation dans les bases test exclusivement, afin de garantir l'indépendance avec les bases d'entraînement.

Pour constituer les bases d'apprentissage, des images ont été tirées au hasard parmi les images déjà triées manuellement (en respectant la distribution entraînement/test définie au paragraphe précédent). Le tirage s'est fait de manière pseudo-aléatoire, avec une **graine aléatoire** (*random seed*) permettant de reproduire les résultats à l'identique. Un **prétraitement** a également été effectué sur toutes les images : elles ont d'abord été recadrées pour ne conserver que la partie de l'image située à gauche du panneau de signalisation (fig. 26), puis redimensionnées en 224×224 pour avoir bonne taille à l'entrée du VGG. [25]

Figure 26 – Recadrage d'une image 1280x720 en 940x720 pour ne conserver que la partie à gauche du panneau

25. Ce recadrage a été fait pour éviter les situations difficilement lisibles sur le bord droit de l'image, où on voit souvent plusieurs types de véhicules au même endroit, qui peuvent aussi être partiellement cachés par le panneau.

Bases de données d'apprentissage pour le classifieur noPL/PL
La base d'entraînement **noPL/PL** (table 1) a servi à entraîner un classifieur binaire afin accélérer le tri manuel des images (en repérant d'emblée les images qui ne contiennent pas de poids lourd). Elle est constituée de deux grandes parties pour un total de 55 547 images :

- Une partie "**noPL**" (absence de poids lourd, environ 40% des images) : ces images proviennent toutes de la classe "V.Inexploitable".
- Une partie "**PL**" (présence de poids lourd, environ 60% des images) : ces images proviennent des autres classes : Bus, Grumiers, TMDs, Autres camions

Classes binaires	**PL**				**noPL**
Sous-classes	Bus	Grumiers	TMDs	Autres camions	V.Inexploitable
Nombre d'images	1724	10449	2738	17458	23178
Pourcentage	3.1%	18.8%	4.9%	31.4%	41.7%

Table 1 – Base d'entraînement pour le classifieur binaire noPL/PL

La base de données de test associée (table 2) a la même structure, mais avec moins d'images (10% du volume total des deux bases) :

Classes binaires	**PL**				**noPL**
Sous-classes	Bus	Grumiers	TMDs	Autres camions	V.Inexploitable
Nombre d'images	149	1126	276	1854	2765
Pourcentage	2.4%	18.2%	4.5%	30.0%	44.8%

Table 2 – Base de test pour le classifieur binaire noPL/PL

Bases de donnée d'apprentissage pour le classifieur TMD/noTMD
La base d'entraînement **TMD/noTMD** (table 3) a été une expérience préliminaire à la mise en œuvre d'un détecteur de plaques TMD à proprement parler. Plus petite que la première (car les images de TMD sont plus rares), elle est constituée de deux parties pour un total de 5 474 images :

- Une partie "**TMD**" (présence de TMD, 50% des images) : ces images proviennent toutes de la classe "TMD".
- Une partie "**noTMD**" (absence de TMD, 50% des images) : ces images proviennent des autres classes : Bus, Grumiers, Autres camions, V.Inexploitable.

Afin d'éviter une surreprésentation des images de la classe V.Inexploitable, le choix a été fait d'équilibrer cette base en prenant comme référence le nombre d'images de TMD : chacune des quatre classes non TMD compte pour un quart de toutes les images de TMD.

Classes binaires	**noTMD**				**TMD**
Sous-classes	Bus	Grumiers	Autres camions	V.Inexploitable	TMDs
Nombre d'images	684	684	684	684	2738
Pourcentage	12.5%	12.5%	12.5%	12.5%	50.0%

Table 3 – Base d'entraînement pour le classifieur binaire TMD/noTMD

La base de données de test associée (table 4) a la même structure, mais avec moins d'images (10% du volume total des deux bases) :

Classes binaires	**noTMD**				**TMD**
Sous-classes	Bus	Grumiers	Autres camions	V.Inexploitable	TMDs
Nombre d'images	69	69	69	69	276
Pourcentage	12.5%	12.5%	12.5%	12.5%	50.0%

Table 4 – Base de test pour le classifieur binaire TMD/noTMD

Base de test applicative pour les deux classifieurs
Cette base table 5 est la réunion de toutes les images des journées réservées au test. Son contenu est représentatif (même répartition des classes d'images, des conditions d'éclairage...) des volumes de données que l'on souhaite traiter en conditions réelles. Cette base permet donc d'avoir une idée plus précise de la précision des classifieurs, qui dépend beaucoup de l'équilibrage des classes. La liste des 10 journées utilisées est donnée en annexe (section 8).

Classe	**Bus**	**Grumiers**	**Autres camions**	**V.Inexploitable**	**TMDs**
Nombre d'images	233	2539	12160	166156	431
Pourcentage	0.1%	1.4%	6.7%	91.5%	0.2%

Table 5 – Base de test applicative (images de 10 journées test complètes)

4.2.3 Entraînement et évaluation des modèles

Pour entraîner le VGG16 modifié, une approche de *transfer learning* a été utilisée : seules les nouvelles couches que nous avons ajoutées ont été entraînées, les poids des autres couches sont restés inchangés (identiques à ceux du VGG16 original). Au total, 4 196 353 paramètres sur les 138 456 897 paramètres (soit environ 10%) ont été entraînés. Cette approche permet d'accélérer significativement l'entraînement. Elle permet aussi de réduire la complexité du modèle, ce qui limite le phénomène de sur-apprentissage.

Les entraînements ont été faits sur GPU avec les hyperparamètres suivants :

- Fonction de coût (*loss*) à optimiser : cross-entropie binaire
- Optimiseur : Adam [23]
- Taille de batch : $batch\_size = 128$
- Taux d'apprentissage (*learning rate*) : $lr = 0.001$
- Utilisation d'une fonction (*callback* `ReduceLROnPlateau`) pour mettre à jour le *learning rate* pendant l'apprentissage : cette fonction a été configurée pour diviser par deux le *learning rate* toutes les 25 époques si la *loss* n'a pas assez diminué.

Après l'entraînement des deux modèles de classification, leur évaluation sur les bases de test est faite en deux étapes. La première est l'**inférence** sur les données : les modèles sont appliqués à toutes les images des bases de test, et les scores obtenus (entre 0 et 1) sont sauvegardés en brut sous forme d'un fichier TXT, où chaque ligne contient le nom d'une image et le score et le score de prédiction associé. La deuxième étape est le calcul des **métriques**

d'évaluation (rappel, précision, score F1). Pour plusieurs seuils de confiance, on compare les prédictions avec les véritables classes des images test pour obtenir les mesures associées à ce score de confiance. Les résultats de cette évaluation seront présentés dans la section 5.2.

4.3 Détection de plaques ADR

Si les classifieurs binaires peut être utiles pour faciliter le tri des images, ces outils ne permettent pas de détecter efficacement les véhicules transportant des matières dangereuses à proprement parler. En effet, la silhouette d'un TMD peut être trompeuse : si la plupart ont le gabarit d'un camion citerne, certains sont aussi des camions ou des véhicules plus légers (fig. 27b). Réciproquement, beaucoup de camions citerne ne sont pas des TMD (fig. 27a). C'est pourquoi l'objectif au cœur de mon stage a été l'entraînement d'un modèle de détection des plaques TMD, capable de les localiser spécifiquement dans les images en les entourant avec des boîtes englobantes.

(a) Un camion citerne ne transportant pas de matières dangereuses

(b) Un camion ordinaire transportant des matières dangereuses

Figure 27 – Exemples de silhouettes de véhicule trompeuses pour les classifieurs binaires

4.3.1 Mise en œuvre d'un réseau RetinaNet

Le détecteur mis en œuvre pendant mon stage est le RetinaNet ([40], voir section 3.4.1), car c'est un réseau récent et performant. Il a été utilisé par l'équipe ENDSUM pour la détection automatique de plaques d'immatriculation et de visages et avait donné de bons résultats [47]. Une implémentation fournie par les auteurs de la publication et disponible sur un dépôt GitHub ([48]) a été utilisée. Celle-ci utilise les librairies Keras (2.4.0) et Tensorflow (2.3.0) de Python. Comme pour le VGG16, je l'ai installée dans un environnement Anaconda dédié.

Cette implémentation permet de spécifier l'architecture du modèle par deux moyens :

- Un fichier de configuration `config.ini` permet de choisir notamment les proportions (*ratios*) et les échelles (*scales*) à utiliser pour les ancres. Pour ce projet, j'ai conservé les paramètres proposés par défaut par les développeurs du RetinaNet.
- Lors de l'entraînement, un paramètre *backbone* permet de choisir le réseau à utiliser pour la partie convolutive du RetinaNet. J'ai gardé l'option par défaut qui est une architecture ResNet [34].

4.3.2 Bases de données d'entraînement et de test

Pour entraîner et évaluer le détecteur RetinaNet, un total de cinq bases de données ont été utilisées :

- Quatre bases d'entraînement *d'apprentissage* (entraînement+validation) et leurs bases de test *d'apprentissage* associées (70% entraînement, 15% validation, 15% test). Elles ont été construites avec une complexité incrémentielle, en incorporant d'abord des exemples positifs (images avec présence de plaque TMD), puis des exemples négatifs "courants", et enfin des exemples négatifs "difficiles" (faux positifs sélectionnés suite à une inférence avec un RetinaNet entraîné) [26]

- Une base de test *applicative* constituée de la totalité des images issues d'une journée (17 mai 2022). Comme pour la classification, cette base est importante pour mesurer l'efficacité du modèle en conditions "réelles", où la distribution des différents types de véhicules n'est pas la même qu'en conditions d'apprentissage. Pour garantir l'indépendance de cette base, les images de cette journée ont été exclues de toutes les bases d'apprentissage.

Pour constituer ces bases d'apprentissage, des images ont été tirées au hasard parmi les images de TMDs annotées manuellement, les exemples négatifs "courants" et les exemples négatifs "difficiles". Comme pour la classification, une graine aléatoire a été utilisée pour permettre la reproduction des bases de données à l'identique. Les images ont aussi été redimensionnées pour que les proportions des objets soient les mêmes [27]

En générant les bases de données, seul le premier niveau d'information des annotations a été conservé. Ainsi, les bases d'apprentissage et la base applicative contiennent uniquement trois classes d'objets :

- `arriere_adr_symb` : Toutes les plaques étiquettes (losanges affichant un symbole) à l'arrière des véhicules

- `arriere_adr_code` : Toutes les plaques de code danger (rectangles affichant un numéro) à l'arrière des véhicules

- `arriere_adr_code_orange` : Toutes les plaques rectangulaires orange (n'affichant aucun numéro) à l'arrière des véhicules [28]

Enfin, pour fournir les données d'entraînement au RetinaNet, il a aussi été nécessaire de générer des fichiers CSV listant le contenu de chaque base de données. Chaque ligne de ces fichiers correspond à une boîte englobante. Elle contient le chemin d'une image, les coordonnées de la boîte et l'identifiant de la classe d'objet associée. Les exemples négatifs (sans annotations) sont listés sur une seule ligne contenant uniquement le nom de l'image.

Bases d'apprentissage

Quatre modèles RetinaNet ont été entraînés sur des bases d'apprentissage d'une complexité croissante. Ces bases ont été constituées à partir de trois collections d'images :

- Une collection de 1979 images de TMD annotées à la main (exemples positifs)

- Une collection d'images de véhicules non TMD prélevées au hasard parmi toutes les images triées (exemples négatifs courants). Collection comptant un peu moins de 4000 images dans chacune des catégories non TMD (bus, grumiers, autres camions, inexploitable)

26. Tous ces exemples négatifs ont dû être contrôlés pour s'assurer qu'aucune plaque TMD n'est présente dans les images

27. Toutes les images n'ont pas le même format, voir annexe section 8

28. Le choix a été fait d'isoler cette classe car elle revêt une signification particulière. En effet, une plaque orange sans code peut signaler le transport de diverses matières dangereuses.

- Une collection d'images de véhicules non TMD susceptibles de mettre le RetinaNet en difficulté (exemples négatifs difficiles). Ces images ont été sélectionnées parmi les faux positifs obtenus lors d'une inférence avec un premier prototype de détecteur. Collection comptant environ 600 images de Bus, 3000 images de grumiers, 2000 images d'autres camions et 1300 images inexploitables.

Quatre bases de données ont été créées, avec 70% d'images pour l'entraînement, 15% d'images de validation et 15% d'images test.

- Base d'apprentissage **"TMD"** (**DB1**) : contient uniquement des exemples positifs
- Base d'apprentissage **"TMD + noTMD courant"** (**DB2**) : contient les mêmes exemples positifs, et autant d'exemples négatifs courants
- Base d'apprentissage **"TMD + noTMD difficile"** (**DB3**) : contient les mêmes exemples positifs, et autant d'exemples négatifs difficiles
- Base d'apprentissage **"TMD + noTMD courant + noTMD difficile"** (**DB4**) : contient les mêmes exemples positifs, autant d'exemples négatifs courants et autant d'exemples négatifs difficiles

Les exemples négatifs sont toujours équitablement distribués entre les quatre classes "Bus", "Grumiers", "Autres camions" et "Inexploitable". La composition de ces bases de données est détaillée dans le tableau 4.3.2.

		TMD	noTMD "courant"	noTMD "difficile"
DB1	**Train**	1385	0	0
	Validation	297	0	0
	Test	297	0	0
DB2	**Train**	1385	1380	0
	Validation	297	296	0
	Test	297	300	0
DB3	**Train**	1385	0	1380
	Validation	297	0	296
	Test	297	0	300
DB4	**Train**	1385	1380	1380
	Validation	297	296	296
	Test	297	300	300

Table 6 – Bases d'entraînement pour le RetinaNet

Base de test applicative (journée test du 17 mai 2022)
Les images d'une journée test (17 mai 2022) ont spécifiquement été annotées pour permettre l'évaluation des détecteurs avec une distribution des données plus "naturelle". Cette base compte 48 177 images sans TMD, et 504 images de TMD annotées (environ 1% des images). En termes de séquence vidéo, cela correspond à 19 séquences vidéos avec un TMD et 640 séquences sans TMD.

4.3.3 Entraînement des modèles

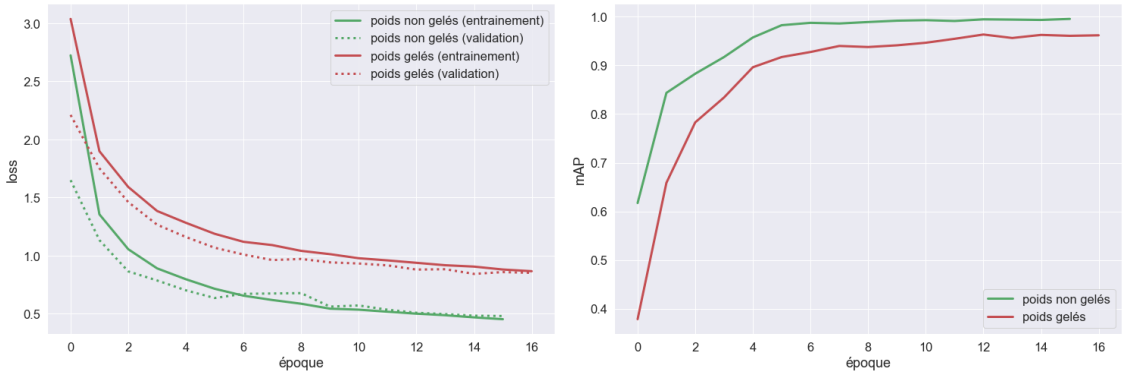
Pour entraîner le RetinaNet, un script d'entraînement fourni par les développeurs du modèle a été utilisé. Celui-ci permet de spécifier les bases de données d'entraînement et

de validation et de choisir les hyperparamètres pour l'entraînement (taille de batch, taux d'apprentissage...). Il est possible d'utiliser un modèle avec des poids préentraînés. Ainsi, les poids optimaux du ResNet [34] entraîné sur la base de données COCO ont été utilisés. Plus d'informations sur l'usage de ce script sont données en annexe.

Les entraînements ont été faits sur GPU avec les hyperparamètres suivants :

- Fonction de coût (*loss*) à optimiser : somme des loss de régression (L1) et de classification (Focal Loss) des ancres
- Optimiseur : Adam [23]
- Taille de batch : $batch\_size = 5$
- Taux d'apprentissage (*learning rate*) : $lr = 10^{-5}$ (par défaut)

Enfin, la question s'est posée de figer ou non les poids de la partie convolutive du RetinaNet pendant l'entraînement (*transfer learning*). En comparant les courbes d'entraînement sur la première base de données **"TMD"** avec et sans transfert fig. 28, on remarque que le modèle arrive à de meilleures performances sans transfert de poids. Tous les entraînements ont donc été faits sans *transfer learning*.



(a) Loss (régression+classification) (b) mAP

Figure 28 – Entraînement du RetinaNet avec et sans gel des poids du ResNet

4.3.4 Algorithme d'évaluation

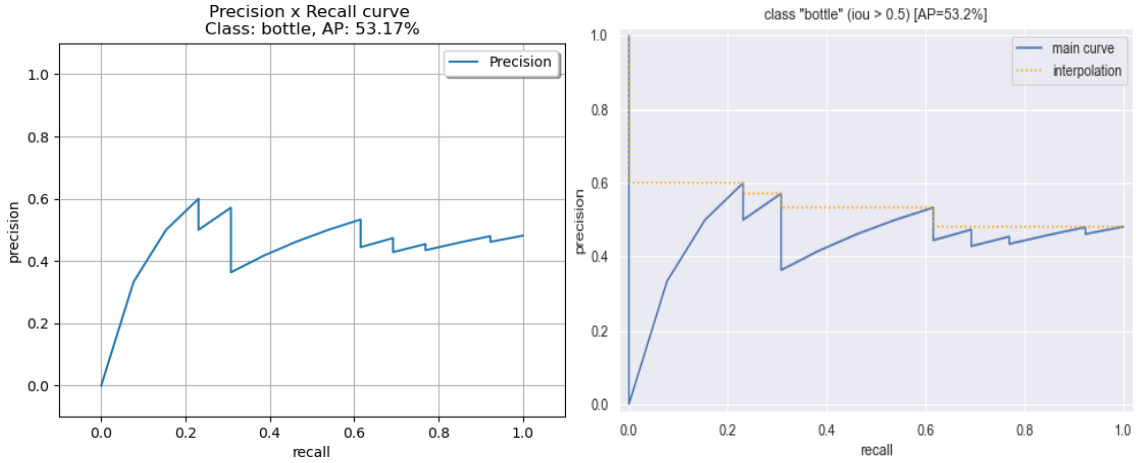
Comme pour le VGG16, l'évaluation de RetinaNet a été faite après avoir enregistré en brut les résultats de toutes les inférences sur les images des bases de test.

Pour chaque classe d'objet à détecter, le **rappel**, la **précision** et le **score F1** ont été calculés en fonction des scores de confiance du modèle. Puis, ces métriques ont été intégrées pour calculer les précisions moyennes de chaque classe, ou **AP** (*average precision*). Enfin, la moyenne des précisions moyennes, ou **mAP** (*mean average precision*) a été calculée pour résumer la performance des modèles en une seule métrique. Toutes ces métriques ont été calculées avec un seuil d'**IOU** égal à 0.5 (une détection est considérée comme un vrai positif si son IOU avec la boîte de la vérité terrain correspondante dépasse le seuil).

Pour calculer ces métriques, j'ai utilisé dans un premier temps un outil développé par les auteurs de [43] appelé *Object Detection Metrics*, mis à disposition sur un dépôt GitHub [11]. Cet outil implémente les méthodes décrites dans la publication et permet d'obtenir rapidement les précisions moyennes, ainsi que la courbe de la précision en fonction du rappel.

Cependant, pour les besoins de ce projet, nous souhaitions aussi pouvoir tracer les courbes de rappel, de précision et de score F1 en fonction des différents niveaux de confiance. Ces courbes n'étant pas fournies directement par *Object Detection Metrics*, j'ai rédigé un script d'évaluation en langage Python pour les implémenter. Ce script calcule les différentes métriques d'une manière proche de celle décrite dans [43] (voir section 3.4.2), mais un changement a dû être apporté dans l'algorithme calculant les rappels et précisions. En effet, la méthode originale donne la courbe précision/rappel, mais chaque point de cette courbe correspond à une unique détection, et non à un score de confiance [29]. Il n'est donc pas possible d'isoler la courbe de rappel, ni la courbe de précision ou de score F1, qui doivent être tracées en fonction du seuil de confiance.

L'algorithme que j'ai implémenté calcule séparément les courbes de précision et de rappel en fonction des niveaux de confiance dans un premier temps. Puis ces métriques sont combinées pour obtenir la courbe précision/rappel, qui est enfin intégrée pour obtenir la précision moyenne [30]. En développant ce script, j'ai utilisé des exemples fournis avec le programme *Object Detection Metrics* pour m'assurer que les courbes et les métriques obtenues soient les plus proches possibles. La figure 29 montre une comparaison entre les courbes de précision/rappel et les précisions moyennes obtenues avec *Object Detection Metrics* et avec le script que j'ai développé. L'aspect des deux courbes est très proche et la précision moyenne (AP) quasi identique. J'ai obtenu une grande similitude des résultats pour 20 des 21 classes d'exemples fournies, ce qui m'a permis de valider ma méthodologie d'évaluation.



(a) *Object Detection Metrics* (**AP**=53.17%)

(b) Algorithme implémenté pour ce stage (**AP**=53.2%)

Figure 29 – Comparaison des courbes de précision/rappel pour la détection d'objets, obtenues avec deux algorithmes différents

29. plusieurs détections peuvent en effet avoir le même score de confiance

30. La méthode d'interpolation pour l'intégration est la même que dans [43] : à chaque valeur de rappel on associe le maximum des précisions à sa droite

5 Résultats

5.1 Statistiques sur les données

Au terme de mon stage, la base de données dont nous disposons compte 2 048 123 images triées en cinq catégories. Ce tri a d'abord été fait entièrement à la main, avant d'être assisté par le modèle de classification binaire **noPL/PL** que j'ai entraîné. Plus précisément, on compte :

- 4 059 images de bus
- 44 432 images de grumiers
- 7 899 images de TMDs
- 201 114 images d'autres camions
- 1 790 619 images classées en inexploitable

La figure 22 présente leur répartition.

De plus, 3 971 images de TMDs ont été annotées à la main pour entraîner et évaluer le détecteur RetinaNet (2 055 d'entre elles ont été utilisées dans les bases d'apprentissage et 504 dans la base de test applicative, les 1 543 images restantes ont été annotées à la fin du stage et sont encore à incorporer dans les bases de données d'apprentissage ou de test).

Afin de mieux comprendre les données annotées, j'ai effectué des statistiques sur les boîtes englobantes des images utilisées pour l'apprentissage. Ces résultats peuvent servir à prendre des décisions pour de futurs entraînements ou pour faire du post-traitement sur les prédictions du RetinaNet. Elles peuvent aussi servir pour essayer d'autres configurations du détecteur (ratios et échelles des ancres).

Une première statistique intéressante consiste à superposer sur une même image toutes les boîtes englobantes annotées à la main, afin de mettre en évidence les endroits de l'image où des plaques TMD sont le plus susceptible d'être visibles. Le résultat d'une telle superposition est visible figure 30.
On remarque que les plaques TMD sont situées dans un faisceau, ce qui peut permettre de filtrer de mauvaises détections qui ne seraient pas à la bonne place dans l'image, ou dont la taille serait incohérente avec la position.

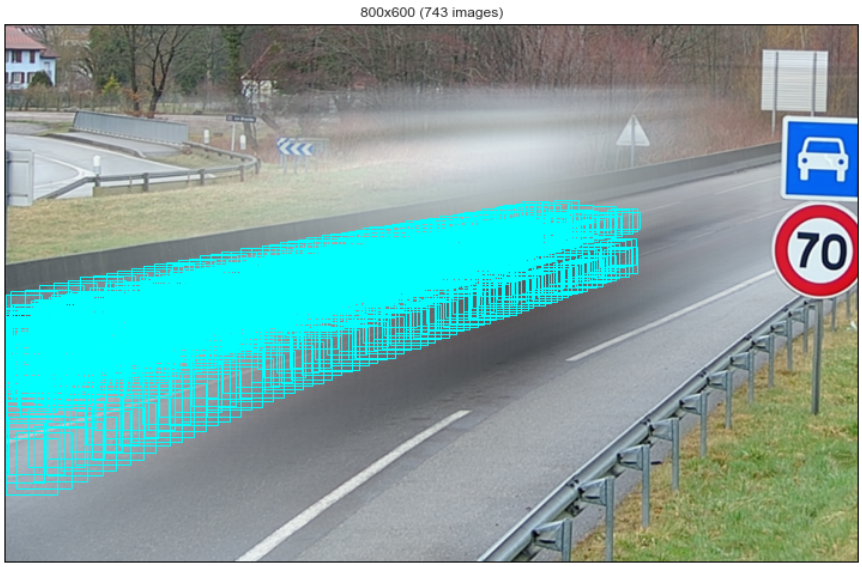


Figure 30 – Superposition de boîtes englobantes sur les images de TMD annotées à la main.

La deuxième statistique effectuée porte sur les propriétés des boîtes englobantes à proprement parler. Elle a été faite en regroupant les labels en trois catégories comme lors de l'entraînement du RetinaNet section 4.3.2 (`arriere_adr_symb`, `arriere_adr_code`, `arriere_adr_code_orange`). Les figures 31, 32 et 33 présentent, pour chacune des trois classes de boîtes englobantes, les distributions de leurs propriétés géométriques (hauteur, largeur, ratio, aire) et colorimétriques (teinte, saturation et luminance moyennes). Les histogrammes de ces grandeurs y sont représentés, accompagnés d'une estimation de leurs densités de probabilité (calculées avec une estimation par noyau, ou *kernel density estimation*).
Sur ces statistiques, on remarque notamment un grand nombre de boîtes de petite aire. Leur présence est due au choix d'annotation à toutes les échelles. Les ratios moyens des boîtes permettent de s'assurer que tous les objets annotés ont les bonnes proportions : on retrouve des ratios proches de 1 et de $4:3$ pour les losanges et les rectangles respectivement. Les statistiques de colorimétrie sont plus difficiles à interpréter, en particulier la teinte des losanges qui varie beaucoup car les couleurs des symboles sont variées (principalement blancs et rouges) et les coins de leurs boîtes englobantes (qui occupent la moitié de la surface) ont la couleur du véhicule, ce qui contribue à bruiter encore plus la distribution. En revanche, la teinte des rectangles est fortement influencée par leur fond orange.

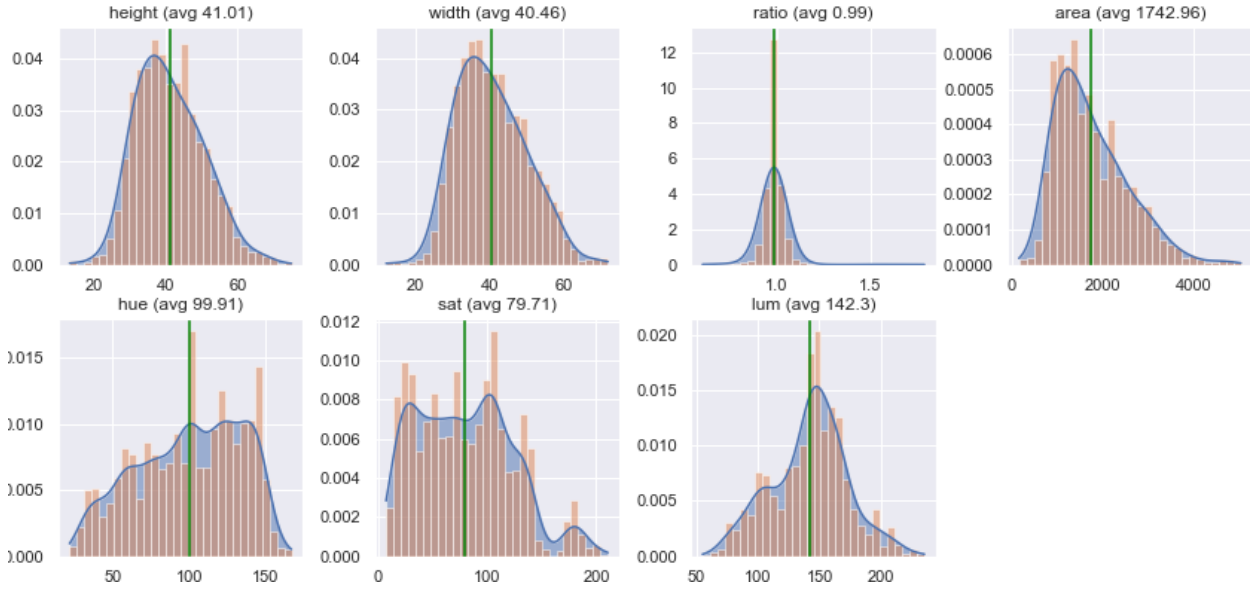


Figure 31 – Statistiques des boîtes `arriere_adr_symb` (2308 boîtes)

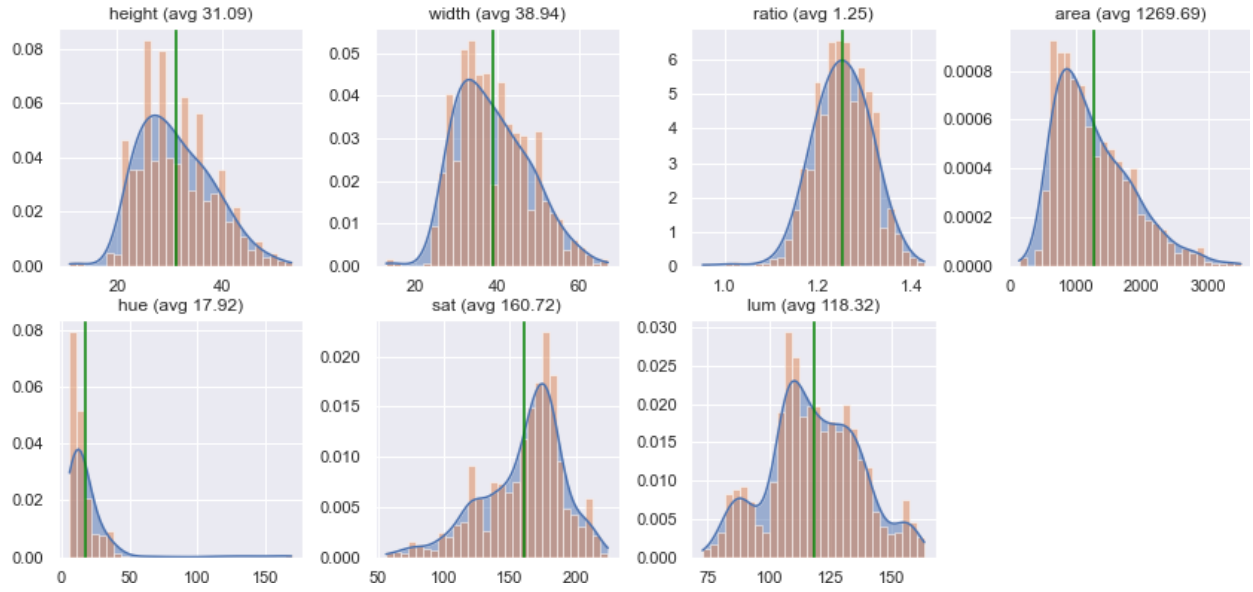


Figure 32 – Statistiques des boîtes `arriere_adr_code` (1012 boîtes)

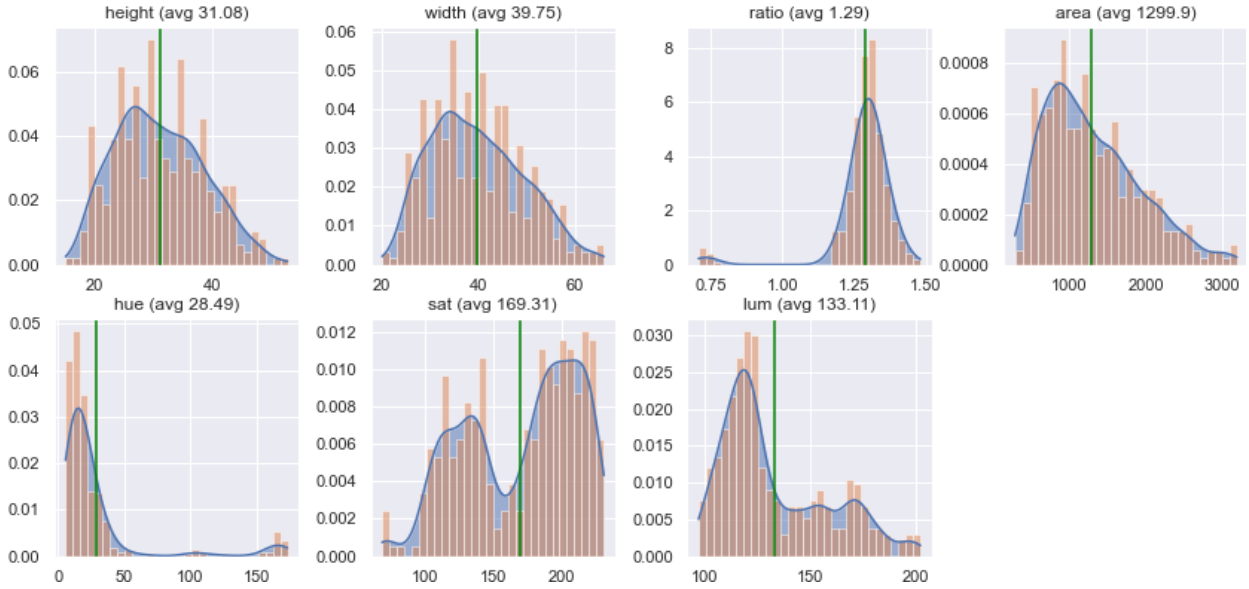


Figure 33 – Statistiques des boîtes `arriere_adr_code_orange` (382 boîtes)

5.2 Évaluation des modèles de classification (VGG16)

L'entraînement du VGG16 sur les bases d'apprentissage a permis d'obtenir deux modèles de classification binaire (**noPL/PL** et **TMD/noTMD**). Chacun est encapsulé dans un fichier au format `H5` pesant 560 Mo. Leur temps d'inférence moyen sur CPU est d'environ 4.4 images par seconde.

Je présenterai dans un premier temps l'évaluation du classifieur binaire **noPL/PL** (absence de poids lourd / présence de poids lourd) puis dans un second temps l'évaluation du classifieur **TMD/noTMD** (présence de TMD / absence de TMD). Chacune de ces évaluations se fera d'abord sur une base de test d'apprentissage, puis sur une base de test applicative.

Afin de permettre une meilleure lisibilité des courbes, les valeurs affichées en abscisse et en ordonnée ne vont pas de 0 à 1, mais du minimum au maximum des valeurs disponibles.

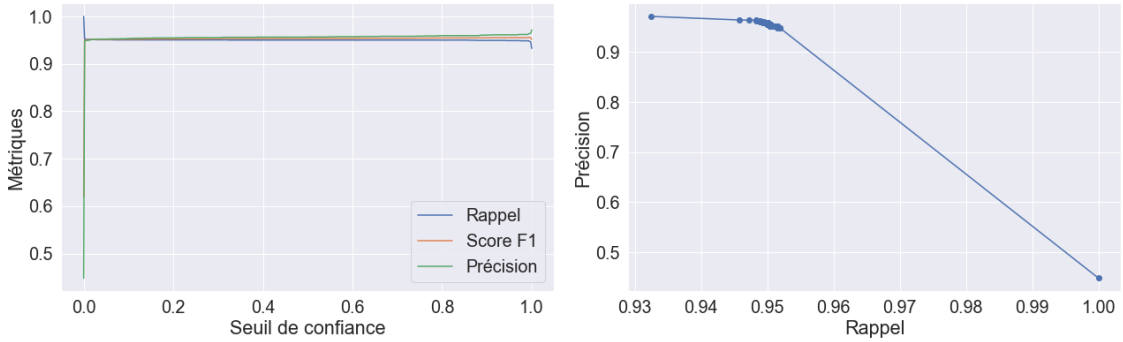
5.2.1 Évaluation du classifieur binaire noPL/PL

Le classifieur **noPL/PL** (absence de poids lourd / présence de poids lourd) a été évalué sur sa base de test d'apprentissage (table 2) et sur la base de test applicative commune aux deux classifieurs (table 5).

Métriques noPL/PL pour la base de test d'apprentissage
Pour la base de test d'apprentissage (table 2), les courbes de précision, rappel et score F1 en fonction du seuil de confiance sont visibles fig. 34a, et la courbe précision/rappel associée est tracée fig. 34b. Le maximum du score F1 est atteint ponctuellement au seuil de confiance 0.99. Avec ce seuil, on obtient pour la classe **noPL** un rappel de 94.8% et une précision de 96.3%, qui sont de bonnes performances. On constate cependant que ces courbes contiennent peu d'information, car les prédictions du modèle sont très polarisées, c'est-à-dire très proches de 0 ou très proches de 1. Ainsi, le choix d'un seuil de décision n'a quasiment aucune influence sur la précision et le rappel de ce modèle.

J'ai aussi observé les performances du classifieur sur cette base d'apprentissage en fixant arbitrairement un seuil de décision à 0.95 (privilégiant légèrement la précision par rapport au

rappel). La matrice de confusion et les autres métriques calculées avec ce seuil de décision sont détaillées fig. 35). Ces performances sont bonnes, avec tous les indicateurs supérieurs à 94%. Cependant, elles doivent être considérées avec un regard critique. En effet, la distribution des classes noPL (V.Inexploitable) et PL (Bus, Grumiers, TMD, Autres camions) dans cette base de données est globalement équilibrée (environ 40% et 60% respectivement), ce qui ne correspond pas à une situation réelle où la grande majorité des images sont dans la classe noPL.



(a) Précision, rappel et score F1

(b) Courbe précision/rappel

Figure 34 – Performances du classifieur **noPL/PL** évalué sur sa **base de test d'apprentissage** (table 2)

	noPL	PL
Rappel	94.9%	96.9%
Précision	96.1%	95.9%
Score F1	95.5%	96.4%
Accuracy	96.0%	

(a) Performances

Vérité \ Prédiction	noPL	PL
noPL	2624	141
PL	106	3299

(b) Matrice de confusion

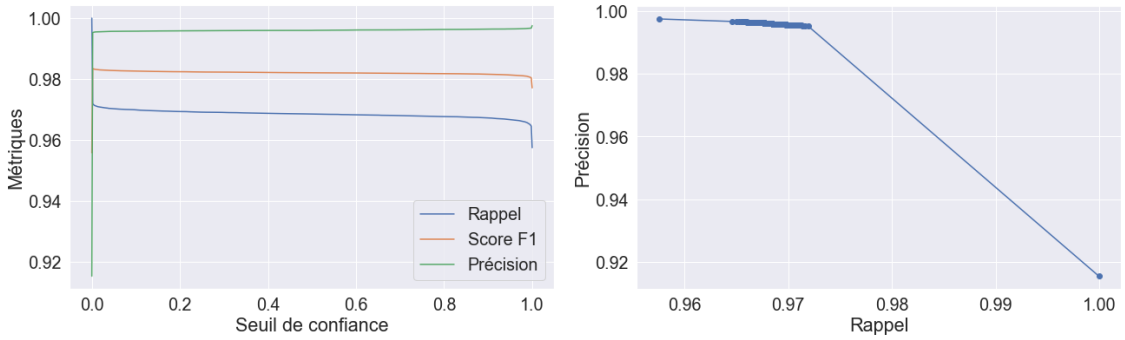
Figure 35 – Performances du classifieur binaire **noPL/PL** évalué sur sa **base de test d'apprentissage** (table 2), pour un seuil de décision de 0.95.

Métriques noPL/PL pour la base de test applicative

Pour la base de test applicative (table 5), les courbes de précision, rappel et score F1 en fonction du seuil de confiance sont visibles fig. 36a, et la courbe précision/rappel associée est tracée fig. 36b. Le maximum du score F1 est atteint ponctuellement au seuil de confiance 0.002 (très faible, alors qu'avec la base d'apprentissage, le seuil optimal valait 0.99). Avec ce seuil, on obtient pour la classe **noPL** un rappel de 97.2% et une précision de 99.5%, étonnamment meilleures que sur la base de test d'apprentissage.

Là encore, les métriques ne varient que très peu en fonction du seuil de décision, car les valeurs des prédictions sont très polarisées. Le score F1 est d'ailleurs décroissant dans ce cas alors qu'il était croissant avec la base d'apprentissage. Ceci explique pourquoi les seuils de confiance optimaux évalués sur la base d'apprentissage et sur la base applicative sont diamétralement opposés. On conclut que le maximum de score F1 n'est pas pertinent pour choisir un seuil de décision dans cette situation. J'ai donc arbitrairement fixé ce seuil à 0.95 (privilégiant la précision par rapport au rappel). La matrice de confusion et les métriques de performances associées sont détaillées fig. 37. Tous les indicateurs de la clase noPL sont supérieurs à 96%, ce qui est une bonne performance. En revanche, la précision pour la classe

PL ne monte plus qu'à 72%. Cela peut s'expliquer par sa sous-représentation (environ 10%) dans la base d'apprentissage par rapport à la classe noPL (environ 90%), ce qui fait que les images noPL mal prédites représentent une part importante des images PL.



(a) Précision, rappel et score F1

(b) Courbe précision/rappel

Figure 36 – Performances du classifieur **noPL/PL** évalué sur la **base de test applicative** du tableau 5

	noPL	PL
Rappel	96.7%	96.3%
Précision	99.6%	72.8%
Score F1	98.1%	82.9%
Accuracy	96.6%	

(a) Performances

Vérité \ Prédiction	noPL	PL
noPL	160622	5534
PL	570	14793

(b) Matrice de confusion

Figure 37 – Performances du classifieur binaire **noPL/PL** évalué sur la **base de test applicative** du tableau 5, pour un seuil de décision de 95%.

5.2.2 Évaluation du classifieur binaire TMD/noTMD

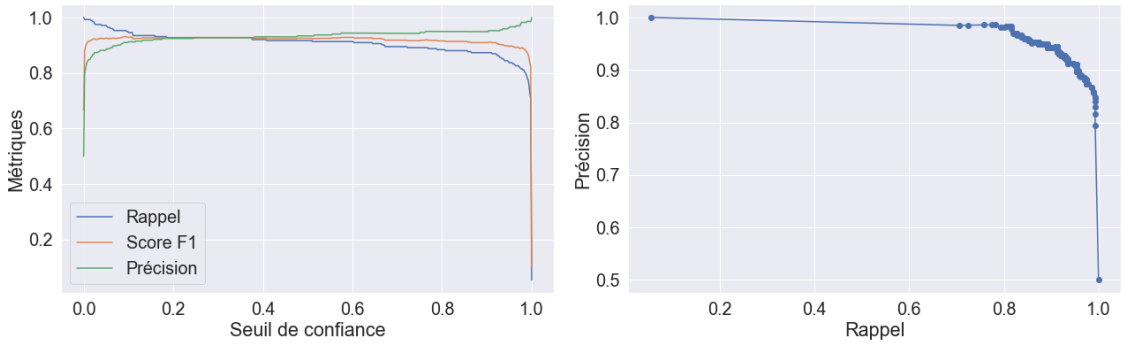
Le classifieur **TMD/noTMD** (présence de TMD / absence de TMD) a été évalué sur sa base de test d'apprentissage (table 4) et sur la base de test applicative commune aux deux classifieurs (table 5).

Métriques TMD/noTMD pour la base de test d'apprentissage

Pour la base de test d'apprentissage (table 4), les courbes de précision, rappel et score F1 en fonction du seuil de confiance sont visibles fig. 38a, et la courbe précision/rappel associée est tracée fig. 38b. Le maximum du score F1 est atteint ponctuellement au seuil de confiance 0.09. Avec ce seuil, on obtient pour la classe **TMD** un rappel de 95.3% et une précision de 91.0%. Ces résultats sont moins satisfaisants que ceux du premier classifieur. Les métriques varient continûment en fonction du seuil de décision, ce qui traduit une distribution moins polarisée des valeurs des prédictions. Ainsi, le choix d'un seuil de décision aura une influence sensible sur la précision et le rappel du modèle.

J'ai observé les performances du classifieur sur cette base d'apprentissage en fixant arbitrairement un seuil de décision à 0.20, privilégiant le rappel par rapport à la précision. En effet, la reconnaissance des TMD est ici prioritaire, on peut donc tolérer une baisse de la précision si cela permet d'éviter des faux négatifs. La matrice de confusion et les autres métriques calculées avec ce seuil sont détaillées fig. 39. Tous les indicateurs restent supérieurs

à 92%, ce qui semblent être une performance correcte. Cependant, ces métriques évaluées dans le contexte de l'apprentissage (avec 50% d'images de TMD, et 50% d'images sans TMD) ne permettent pas de conclure sur les performances qu'aura le modèle en conditions réelles, où les TMD ne représentent qu'une infime partie du trafic routier.



(a) Précision, rappel et score F1

(b) Courbe précision/rappel

Figure 38 – Performances du classifieur **TMD/noTMD** évalué sur sa **base de test d'apprentissage** (table 4)

	TMD	noTMD
Rappel	92.8%	92.4%
Précision	92.4%	92.7%
Score F1	92.6%	92.6%
Accuracy	92.6%	

(a) Performances

Vérité \ Prédiction	TMD	noTMD
TMD	256	20
noTMD	21	255

(b) Matrice de confusion

Figure 39 – Performances du classifieur binaire **TMD/noTMD** évalué sur sa **base de test d'apprentissage** (table 4), pour un seuil de décision de 0.20.

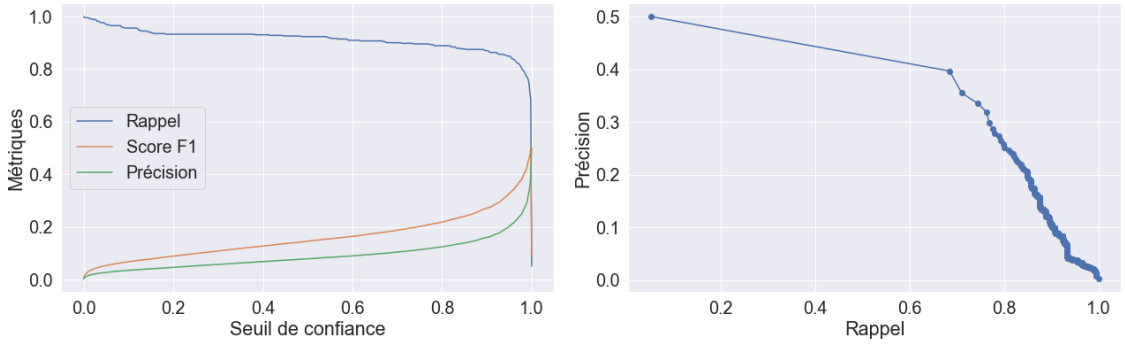
Métriques TMD/noTMD pour la base de test applicative

Pour la base de test applicative (table 5), les courbes de précision, rappel et score F1 en fonction du seuil de confiance sont visibles fig. 40a, et la courbe précision/rappel associée est tracée fig. 40b. Le maximum du score F1 est atteint ponctuellement au seuil de confiance 0.99. Avec ce seuil, on obtient pour la classe **TMD** un rappel de 68.4% et une précision de 39.7%. Contrairement au cas du classifieur noPL/PL, les performances sont ici très fortement dégradées (précision ne dépassant jamais 50%).

Puisque la reconnaissance des TMD sans faux négatifs est un enjeu de sécurité important, le choix d'un seuil de décision faible permet de légèrement progresser sur le rappel du classifieur, mais au prix d'une dégradation encore plus marquée de la précision (faux positifs très nombreux). J'ai donc observé les performances du classifieur en fixant arbitrairement un seuil de décision à 0.20, privilégiant le rappel par rapport à la précision. La matrice de confusion et les métriques associées sont détaillées fig. 41). Le rappel est bon (supérieur à 90%) mais insuffisant, car il est important ici de reconnaître un maximum de TMD. Quant à la précision, elle est très mauvaise (inférieure à 5%), ce qui rend ce modèle inutilisable pour l'application souhaitée.

Des performances si mauvaises pour la reconnaissance des TMD peuvent s'expliquer pour deux raisons :

- Premièrement, le déséquilibre des classes (TMD contre le reste du monde) est ici critique. Il y a si peu d'images de TMD parmi toutes les images collectées (1%, voire moins) que le nombre de faux positifs sera toujours immense comparé au nombre de vrais positifs.
- Deuxièmement, la classification d'une image toute entière en utilisant les caractéristiques de la silhouette d'un véhicule peut mener à des erreurs. Bien que la plupart des TMD circulant au tunnel de Schirmeck soient des camions-citerne, la réciproque n'est pas vraie. De plus, certains TMD sont des camions ordinaires, ou parfois même des véhicules légers. La figure fig. 27 illustre ce problème.



(a) Précision, rappel et score F1

(b) Courbe précision/rappel

Figure 40 – Performances du classifieur **TMD/noTMD** évalué sur la **base de test applicative** du tableau 5

	TMD	noTMD
Rappel	93.3%	95.5%
Précision	4.7%	100.0%
Score F1	8.9%	97.7%
Accuracy	95.5%	

(a) Performances

Vérité \ Prédiction	TMD	noTMD
TMD	402	29
noTMD	8158	172930

(b) Matrice de confusion

Figure 41 – Performances du classifieur binaire **TMD/noTMD** évalué sur la **base de test applicative** du tableau 5, pour un seuil de décision de 0.20

5.2.3 Conclusion sur les modèles de classification

Les deux classifieurs binaires ont été entraînés avec deux objectifs différents. Le modèle **noPL/PL** avait pour but de faciliter le tri manuel des images récoltées au tunnel de Schirmeck, en reconnaissant d'emblée les images inexploitables, tandis que modèle **TMD/noTMD** avait pour but de tester la pertinence de la classification pour la reconnaissance des TMD

L'évaluation a révélé des performances satisfaisantes pour le classifieur noPL/PL, qui a par la suite été intégré directement dans notre algorithme de détramage des vidéos afin d'effectuer un premier tri automatique des images. En revanche, les mauvaises performances du classifieur TMD/noTMD invalident totalement l'approche de reconnaissance des TMD au moyen d'une classification ordinaire. L'issue de cette expérimentation a justifié la mise en œuvre d'un modèle de *détection* des plaques TMD, capable de reconnaître ces véhicules sans tenir compte de leur silhouette.

5.3 Évaluation des modèles de détection des plaques TMD (RetinaNet)

Les entraînements du RetinaNet ont permis d'obtenir quatre modèles de détection des plaques TMD. Chacun est encapsulé dans un fichier au format `H5` pesant 140 Mo. Leur temps d'inférence moyen sur GPU est d'environ 5.7 images par seconde. [31]

Je présenterai dans un premier temps quelques résultats d'évaluation des modèles sur les bases de test *d'apprentissage* définies table 6 : chaque modèle ayant été entraîné sur une base différente, il sera évalué sur la base de test correspondante. Pour plus de simplicité, les bases de données seront désignées ainsi :

- **DB1** : base d'apprentissage **"TMD"**
- **DB2** : base d'apprentissage **"TMD + noTMD courant"**
- **DB3** : base d'apprentissage **"TMD + noTMD difficile"**
- **DB4** : base d'apprentissage **"TMD + noTMD courant + noTMD difficile"**

Dans un second temps, je présenterai plus en détails l'évaluation des modèles sur la base de test *applicative* définie 4.3.2 (journée test du 17 mai 2022). Dans ce cas, tous les modèles sont évalués sur la même base. Je détaillerai davantage l'évaluation du dernier modèle (entraîné sur la base **DB4**).

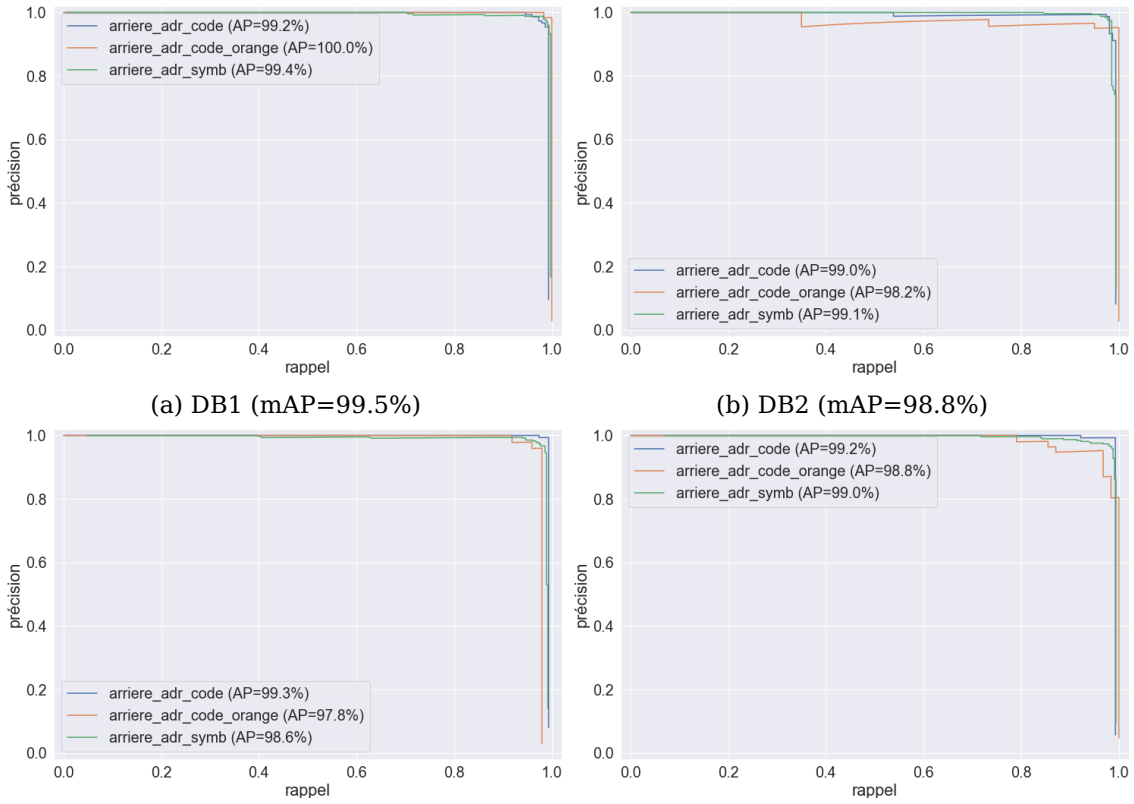
Enfin, je présenterai les résultats d'une première évaluation des modèles qui mettra en lumière leur utilité pour le **comptage des TMD**, qui est l'objectif premier visé par le projet du Cerema en collaboration avec la Communauté européenne d'Alsace.

5.3.1 Évaluation des détecteurs sur leurs bases de test d'apprentissage

Les courbes de précision/rappel des quatre détecteurs évalués sur leurs bases de test d'apprentissage respectives sont visibles fig. 42. À chaque fois, une courbe est tracée pour les trois classes d'objets à détecter (parmi `arriere_adr_symb`, `arriere_adr_code` et `arriere_adr_code_orange`) et sa précision moyenne (AP, *Average Precision*) est mesurée. La moyenne de ces trois mesures (mAP, *mean Average Precision)* est finalement calculée pour résumer la performance du modèle en une seule métrique.

Les performances des modèles sur leurs bases de test d'apprentissage respectives sont toutes excellentes, avec toutes les APs et mAPs supérieures à 97%. Il convient cependant d'être prudent vis-à-vis de ces résultats. Les bases de test d'apprentissage sont tout d'abord très petites, avec quelques centaines d'images seulement. Mais surtout, les images qui les composent proviennent des mêmes séquences vidéo que les images utilisées pour les entraînements, ce qui induit un fort biais lors de l'évaluation. J'ai donc refait l'évaluation des modèles sur la base de test applicative (images de la journée du 17 mai 2022) pour obtenir des métriques plus parlantes. Cette base est véritablement indépendante des données d'entraînement et contient plus d'images, avec une distribution représentative des conditions réelles d'enregistrement des véhicules.

31. Ce temps inclut la création et l'enregistrement d'un fichier d'annotation XML après chaque inférence



(a) DB1 (mAP=99.5%)

(b) DB2 (mAP=98.8%)

(c) DB3 (mAP=98.6%)

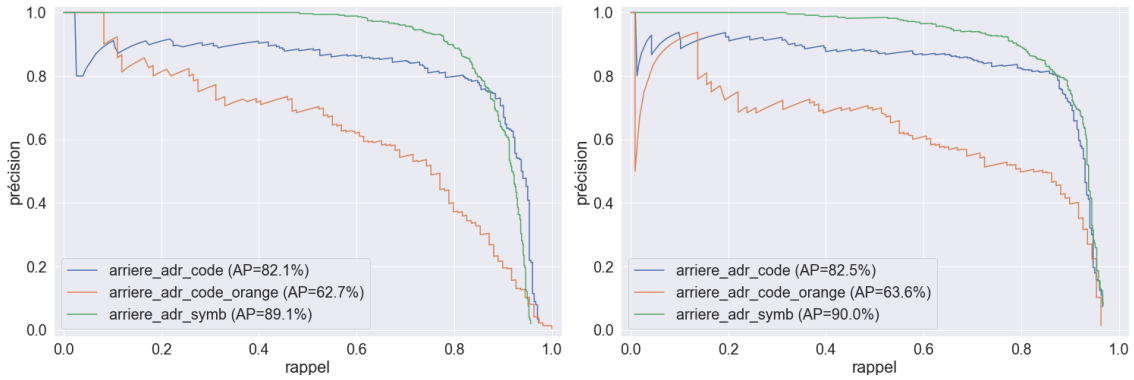
(d) DB4 (mAP=99.0%)

Figure 42 – Courbes de précision/rappel des modèles RetinaNet entraînés et évalués sur quatre bases d'apprentissage de complexité croissante

5.3.2 Évaluation des détecteurs sur une journée test

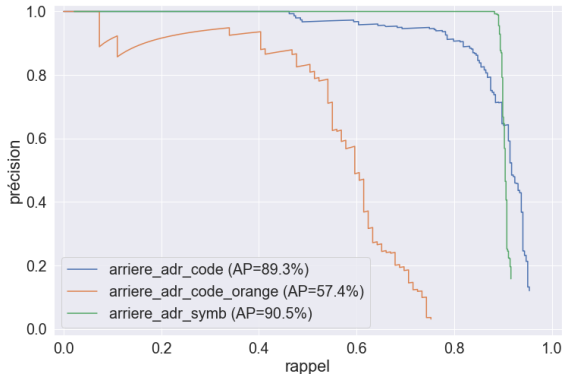
Les courbes de précision/rappel des quatre détecteurs évalués sur la base de test applicative (journée du 17 mai 2022, section 4.3.2) sont visibles fig. 42. Tout comme pour l'évaluation sur les bases de test d'apprentissage, une courbe est tracée pour chacune des trois classes d'objet et sa précision moyenne est calculée. La mAP résume la performance générale des modèles.

Comparatif des performances des quatre modèles Le résultat de ces évaluations sur la même base présente cette fois-ci une progression visible d'un modèle au suivant. La figure 44 présente l'évolution des APs et mAPs d'une base d'entraînement à la suivante. En particulier, on remarque que la mAP générale est strictement croissante. Les précisions moyennes par classe sont aussi croissantes, avec une exception à la troisième base de données ("TMD + noTMD difficile"), où on remarque un recul de la précision de la classe `arriere_adr_code_orange` (panneau orange vierge), qui concorde avec une progression de la précision pour la classe `arriere_adr_code` (panneau orange codifié). Ceci peut suggérer que le modèle a eu tendance à confondre les panneaux oranges vierges avec des panneaux oranges codifiés en apprenant sur cette base de données. Enfin, le rappel maximal visible sur les courbes de la figure 42 semble avoir beaucoup de mal à se rapprocher de 100%. Ceci pourrait être causé par l'annotation des images de la base test, où même les plaques sur des véhicules lointains ont été annotées. Il est possible que ces plaques soient trop petites pour être détectées par le RetinaNet avec sa configuration actuelle.

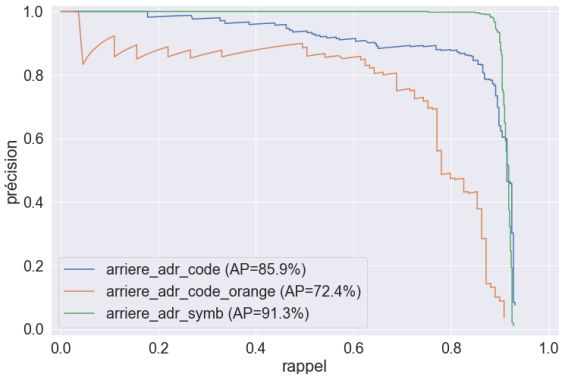


(a) DB1 (mAP=78.0%)

(b) DB2 (mAP=78.7%)



(c) DB3 (mAP=79.0%)



(d) DB4 (mAP=83.2%)

Figure 43 – Courbes de précision/rappel du RetinaNet entraîné sur quatre bases de complexité croissante et évalué sur la **base de test applicative** 4.3.2

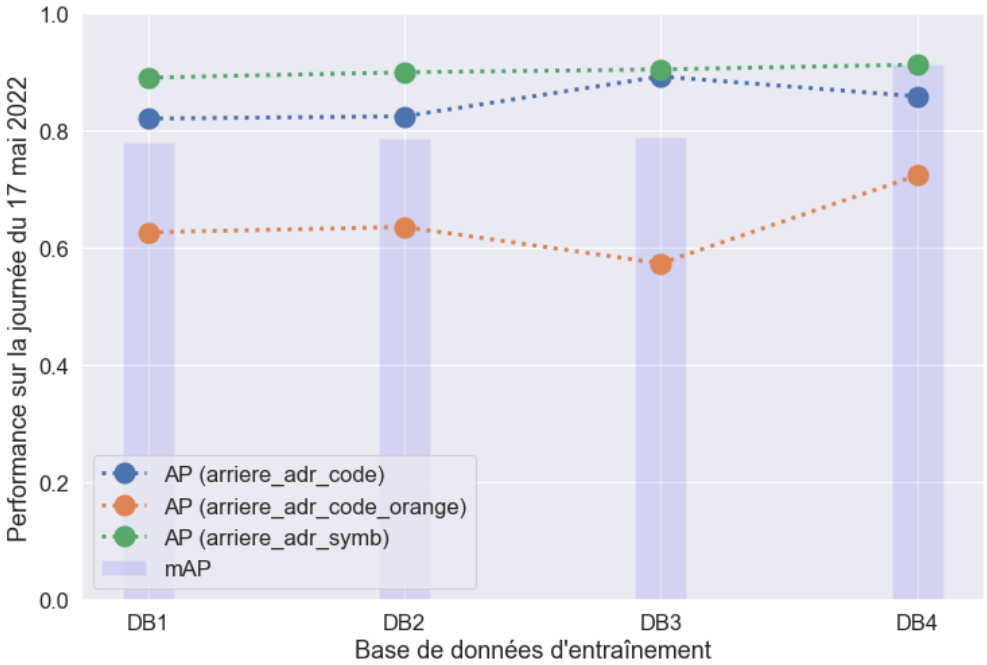
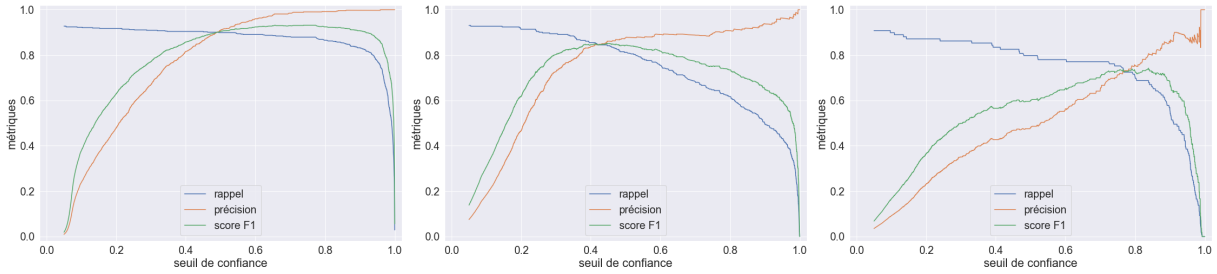


Figure 44 – Progression des performances de détection sur la journée test du 17 mai 2022

Étude des performances du dernier modèle (entraîné sur DB4) Le quatrième détecteur (entraîné sur la base "TMD + noTMD courant + noTMD difficile" affiche la meilleure

précision moyenne globale (mAP). Il est donc intéressant d'observer en détails ses courbes de précision, rappel et de score F1 par classe. La figure 45 présente ces résultats.

On remarque que les seuils de décision optimaux (maximisant le score F1) ne sont pas les mêmes pour chaque classe. Pour les panneaux oranges codifiés, il se situe autour de 0.45 seulement, alors qu'il vaut approximativement 0.85 pour les panneaux oranges vierges et environ 0.70 pour les symboles. La connaissance de ces spécificités doit permettre d'exploiter plus finement le modèle pour détecter les plaques TMD lors de l'inférence sur de nouvelles images.



(a) `arriere_adr_symb` mAP = 91.3%

(b) `arriere_adr_code` mAP = 85.9%

(c) `arriere_adr_code_orange` mAP = 72.4 %

Figure 45 – Précision, rappel et score F1 du RetinaNet pour trois classes d'objet à détecter. Modèle entraîné sur la base **DB4** et évalué sur la **base de test applicative** 4.3.2

5.3.3 Évaluation des détecteurs pour la classification d'images et de vidéos

Jusqu'à présent, les détecteurs de plaques TMD ont été évalués de façon "brute", c'est-à-dire que toutes les boîtes englobantes prédites sont comparées aux boîtes englobantes de la vérité terrain. Dans le cadre du projet en collaboration avec la CeA, on souhaite cependant exploiter les modèles pour le comptage des *véhicules* TMD. C'est pourquoi j'ai également mené une première expérience qualitative pour évaluer cette utilisation du RetinaNet.

Plus spécifiquement, j'ai cherché à utiliser les modèles entraînés de deux façons différentes. D'abord comme **classifieurs binaires d'images** (TMD/noTMD), puis comme **classifieurs binaires de séquences vidéo**. J'ai en effet considéré qu'une vidéo correspond à un véhicule. Sous cette hypothèse, le problème de comptage devient un problème de classification vidéo.[32] J'ai effectué ces deux expériences en utilisant les résultats des inférences sur la journée test du 17 mai 2022 (ces mêmes résultats qui ont servi pour l'évaluation "brute" des modèles sur cette journée dans la section précédente)

Il a été nécessaire de définir deux critères de classification binaire (TMD/noTMD), un premier pour les images, un second pour les vidéos. Pour cette première expérimentation, je me suis contenté de critères très simples :

- **Critère de classification binaire pour les images** : une image est classifiée comme TMD si le modèle y a proposé une boîte englobante avec un score de confiance supérieur à 0.70.
- **Critère de classification binaire pour les vidéos** : une séquence vidéo est classifiée comme TMD si pour au moins une des images le modèle a proposé une boîte englobante avec un score de confiance supérieur à 0.90.

32. En pratique, une séquence vidéo peut couvrir le passage de plusieurs véhicules. Cependant, dans le cadre de cette expérience, il a été facile de vérifier que chaque vidéo de TMD contenait bien un unique TMD.

Avec ces deux critères, j'ai pu faire le comptage des faux positifs et des faux négatifs pour la classification d'image d'une part et pour la classification des séquences vidéo d'autre part.

Les résultats pour la classification d'images sont résumés dans la figure 46. Dans ce contexte, il faut rappeler que la journée test comporte 504 images de TMD (environ 1%) et 48 177 images sans TMD. On observe une diminution progressive du nombre de faux positifs, dans l'ordre "DB1-DB2-DB4-DB3", couplée à augmentation du nombre de faux négatifs. La diminution du nombre de faux négatifs est cependant beaucoup plus rapide que l'augmentation du nombre de faux positifs, ce qui montre la capacité du modèle à apprendre efficacement à partir d'exemples négatifs pour améliorer sa précision sans sacrifier trop de rappel.

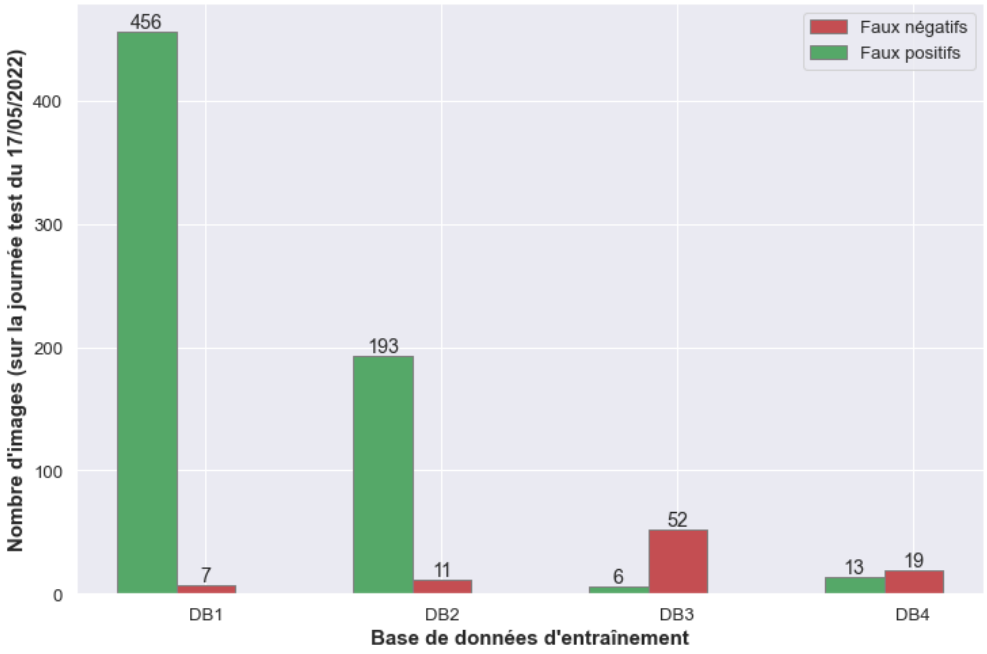


Figure 46 – Faux négatifs et faux positifs pour chaque détecteur **(par image)** lors de l'inférence sur la **base de données applicative** (section 4.3.2). Base comptant 504 images "TMD" et 48 177 images "noTMD"

Les résultats pour la classification de vidéos sont résumés dans la figure 47. Dans ce contexte, il faut rappeler que la journée test comporte 19 vidéos de TMD ($\sim 3\%$) et 640 vidéos sans TMD.[33] Ces résultats sont très prometteurs, car on observe sur cette (petite) base de données qu'aucun TMD n'a été mal classé (aucun faux positif). Un aussi bon résultat est probablement dû à la redondance des images dans une même séquence vidéo, qui donne au détecteur de multiples occasions pour détecter une plaque de TMD. On observe là aussi une diminution importante du nombre de faux positifs d'une base de données à la suivante.[34]

33. Attention, quelques-unes de ces vidéos - moins d'une dizaine - contiennent des TMD circulant dans le sens de circulation opposé, et affichant un panneau orange à l'avant du véhicule. Pour une évaluation plus juste, ces vidéos devraient être considérées comme des exemples positifs dans ce contexte.

34. Suite de la remarque précédente : quelques-uns de ces faux positifs sont en réalité des vidéos avec des TMD circulant en sens inverse, considérés comme des exemples négatifs dans la vérité terrain. Par exemple, 3 des 4 "faux positifs" du modèle entraîné sur DB4 sont en réalité des TMD circulant en sens inverse... qui n'ont pas été détectés par le modèle entraîné sur DB3 !

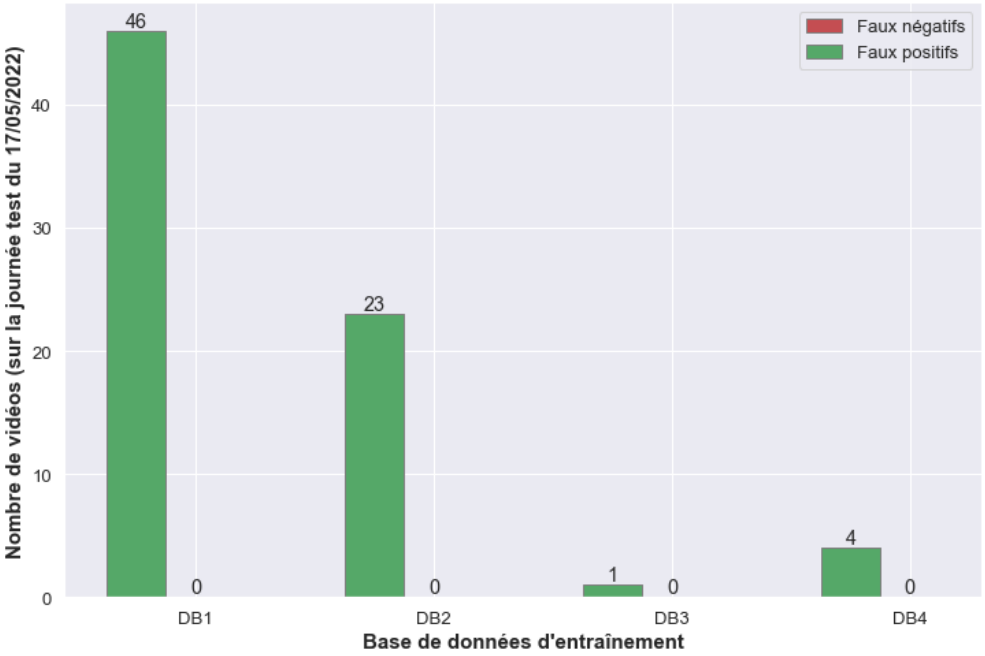


Figure 47 – Faux négatifs et faux positifs pour chaque détecteur **(par séquence vidéo)** lors de l'inférence sur la **base de données applicative** (section 4.3.2). Base comptant 19 vidéos "TMD" et 640 vidéos "noTMD"

5.3.4 Conclusion sur les modèles de détection

Pour conclure sur l'évaluation des modèles de détection, la comparaison des quatre détecteurs entraînés sur des bases d'entraînement de complexité variable a permis de mettre en évidence une progression des performances (sur une journée test) d'un apprentissage au suivant, et ce à plusieurs niveaux d'évaluation :

- Évaluation "brute" (par objet)
- Évaluation dans un contexte de classification d'images
- Évaluation dans un contexte de classification vidéo (important pour le projet de comptage des TMD en partenariat avec la CeA)

La variation des performances suit la même tendance à chaque niveau de l'évaluation, bien qu'il soit un peu plus difficile de départager le troisième et le quatrième modèle.

Les résultats de classification des vidéos sur une journée test, même avec un critère de décision très rudimentaire, sont particulièrement prometteurs.

6 Conclusion et perspectives

Au terme de ces six mois au sein de l'équipe ENDSUM, j'ai pu réaliser l'objectif principal de mon stage, à savoir la mise en œuvre et l'évaluation d'algorithmes de reconnaissance pour l'analyse d'images de trafic routier et la reconnaissance de véhicules transportant des matières dangereuses (TMD).

Deux types d'architectures ont été entraînées : des modèles de classification d'images (VGG16) et des modèles pour la détection des plaques TMD (RetinaNet). L'un des modèles de classification s'est avéré avoir de bonnes performances et a pu être directement exploité pour accélérer le tri des images de notre base de données. De même, les modèles de détections des plaques TMD mis en œuvre ont aussi montré des performances satisfaisantes, avec un potentiel prometteur pour l'application au comptage des TMD (par véhicule). Par ailleurs, l'équipe ENDSUM dispose à présent d'un important corpus d'images triées et annotées pouvant être exploitées pour d'autres projets d'analyse de l'imagerie routière.

Plusieurs pistes d'amélioration peuvent être explorées. En termes de classification, il est possible d'améliorer les bases d'entraînement (nettoyage, ajout de nouvelles données) et d'entraîner des modèles de la classification multiclasse pour accélérer encore plus le tri des images. L'entraînement des modèles peut être accéléré, par exemple en ayant recours à de la *batch-normalisation*[35]. En termes de détection, d'autres politiques d'annotation des images peuvent être explorées afin de trouver un système d'annotation plus efficace pour l'entraînement des détecteurs (choix des labels et de la taille maximale des boîtes englobantes à montrer au modèle, par exemple). Les évaluations sur des bases de données applicatives peuvent être poursuivies en intégrant de nouvelles annotations qui ont été réalisées pendant les dernières semaines du stage. Enfin, l'architecture du RetinaNet en elle-même peut faire l'objet de modifications (ratios et échelles des ancres, choix du modèle à utiliser pour l'extraction des caractéristiques).

De plus, les résultats obtenus au cours de ce stage pourront faire l'objet de futures applications ou déclinaisons opérationnelles, dans le cadre d'un projet en collaboration avec la Collectivité européenne d'Alsace (CeA) visant à développer une application de comptage des TMD. Dans cette optique, un post-traitement des détections proposées par le RetinaNet est souhaitable pour fiabiliser les détections de plaques TMD. Celui-ci pourrait exploiter le contenu, la position, la taille et le mouvement des boîtes englobantes dans les séquences vidéo. Il serait aussi intéressant de mettre en œuvre un outil pour le comptage des véhicules à proprement parler, qui serait capable d'isoler chaque véhicule dans un flux vidéo continu[36]. Enfin, la question de la reconnaissance des plaques TMD reste à explorer : lecture des codes sur les panneaux orange, reconnaissance des pictogrammes visuels…[37]

Enfin, ce stage m'a enrichi personnellement. J'ai pu approfondir les méthodes d'apprentissage profond abordées dans le programme de mon Master, et développer certaines nouvelles habitudes de codage. J'ai également beaucoup apprécié le travail en équipe avec tous les membres de l'équipe ENDSUM.

35. La normalisation par batch consiste à centrer réduire les données de chaque batch en utilisant leur moyenne et leur écart type

36. Actuellement, nous dépendons pour cela du système de déclenchement automatique des caméras installées sur site

37. Il est possible que les méthodes d'apprentissage profond ne soient pas d'une grande aide pour cette analyse, car seule une petite quantité de codes et symboles sont fortement représentés dans nos bases de données. La plupart d'entre eux n'apparaissent que quelques fois, voire pas du tout.

7 Bibliographie

Références

[1] “Les directions et délégations du Cerema | Cerema.” [Online]. Available : https://www.cerema.fr/fr/cerema/directions

[2] UNITED NATIONS, *Accord relatif au transport international des marchandises dangereuses par route*. S.l. : UNITED NATIONS, 2020, oCLC : 1198977571. [Online]. Available : https://unece.org/transportdangerous-goods/adr-2021-files

[3] “Arrêté du 29 mai 2009 relatif aux transports de marchandises dangereuses par voies terrestres (dit « arrêté TMD ») - Légifrance.” [Online]. Available : https://www.legifrance.gouv.fr/loda/id/LEGITEXT000020797782/

[4] “Liste des numéros ONU,” May 2022, page Version ID : 194149969. [Online]. Available : https://fr.wikipedia.org/w/index.php?title=Liste_des_num%C3%A9ros_ONU&oldid=194149969

[5] “Liste des codes Kemler,” May 2022, page Version ID : 193601467. [Online]. Available : https://fr.wikipedia.org/w/index.php?title=Liste_des_codes_Kemler&oldid=193601467

[6] “Accord relatif au transport international des marchandises dangereuses par route,” Sep. 2021, page Version ID : 186290177. [Online]. Available : https://fr.wikipedia.org/w/index.php?title=Accord_relatif_au_transport_international_des_marchandises_dangereuses_par_route&oldid=186290177#Panneaux_Oranges

[7] “Langage Python.” [Online]. Available : https://www.python.org/

[8] “Anaconda Distribution.” [Online]. Available : https://www.anaconda.com/products/distribution

[9] “Tensorflow Keras module.” [Online]. Available : https://www.tensorflow.org/api_docs/python/tf/keras

[10] darrenl, “LabelImg,” May 2022, original-date : 2015-09-17T01 :33 :59Z. [Online]. Available : https://github.com/tzutalin/labelImg

[11] R. Padilla, “Object Detection Metrics,” Jul. 2022, original-date : 2020-11-11T23 :58 :38Z. [Online]. Available : https://github.com/rafaelpadilla/review_object_detection_metrics

[12] “GitHub : Where the world builds software.” [Online]. Available : https://github.com/

[13] “The MIT License | Open Source Initiative.” [Online]. Available : https://opensource.org/licenses/mit-license.php

[14] “GPU Nvidia Quadro RTX 5000.” [Online]. Available : https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/quadro-rtx-5000-data-sheet-us-nvidia-704120-r4-web.pdf

[15] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*, first edition ed. Beijing ; Boston : O’Reilly Media, 2017, oCLC : ocn953432302.

[16] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. [Online]. Available : https://doi.org/10.1007/BF02478259

[17] F. Rosenblatt, “The perceptron : A probabilistic model for information storage and organization in the brain,” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, place : US Publisher : American Psychological Association.

[18] “AlphaGo.” [Online]. Available : https://www.deepmind.com/research/highlighted-research/alphago

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation,” CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, Tech. Rep., Sep. 1985, section : Technical Reports. [Online]. Available : https://apps.dtic.mil/sti/citations/ADA164453

[20] P. Werbos and P. John, “Beyond regression : new tools for prediction and analysis in the behavioral sciences /,” Jan. 1974.

[21] “Chain rule,” Aug. 2022, page Version ID : 1102134961. [Online]. Available : https://en.wikipedia.org/w/index.php?title=Chain_rule

[22] “Règle de Hebb,” Feb. 2022, page Version ID : 191445958. [Online]. Available : https://fr.wikipedia.org/w/index.php?title=R%C3%A8gle_de_Hebb&oldid=191445958

[23] D. P. Kingma and J. Ba, “Adam : A Method for Stochastic Optimization,” Jan. 2017, arXiv :1412.6980 [cs]. [Online]. Available : http://arxiv.org/abs/1412.6980

[24] “[image algorithm] - how does the convolution kernel in convolution neural network (CNN) extract image features (Python realizes image convolution operation),” Dec. 2021. [Online]. Available : https://developpaper.com/image-algorithm-how-does-the-convolution-kernel-in-convolution-neural-network-cnn-extract-image-features-python-realizes-image-convolution-operation/

[25] “Sobel operator,” Aug. 2022, page Version ID : 1104299942. [Online]. Available : https://en.wikipedia.org/w/index.php?title=Sobel_operator&oldid=1104299942

[26] “Softmax function,” Aug. 2022, page Version ID : 1102593845. [Online]. Available : https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=1102593845

[27] F. M. Graetz, “RetinaNet : how Focal Loss fixes Single-Shot Detection,” Dec. 2018. [Online]. Available : https://towardsdatascience.com/retinanet-how-focal-loss-fixes-single-shot-detection-cb320e3bb0de

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2012. [Online]. Available : https://dl.acm.org/doi/10.1145/3065386

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” Sep. 2014, number : arXiv :1409.4842 arXiv :1409.4842 [cs]. [Online]. Available : http://arxiv.org/abs/1409.4842

[30] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” 2014, pp. 580–587. [Online]. Available : https://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html

[31] R. Girshick, “Fast R-CNN,” 2015, pp. 1440–1448. [Online]. Available : https://openaccess.thecvf.com/content_iccv_2015/html/Girshick_Fast_R-CNN_ICCV_2015_paper.html

[32] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks,” in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. [Online]. Available : https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[33] A. Zisserman and K. Simonyan, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv :1409.1556 [cs]*, Apr. 2015, arXiv : 1409.1556. [Online]. Available : http://arxiv.org/abs/1409.1556

[34] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA : IEEE, Jun. 2016, pp. 770–778. [Online]. Available : http://ieeexplore.ieee.org/document/7780459/

[35] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, number : arXiv :1704.04861 arXiv :1704.04861 [cs]. [Online]. Available : http://arxiv.org/abs/1704.04861

[36] “Evaluation measures for multiclass problems.” [Online]. Available : http://gabrielelanaro.github.io/blog/2016/02/03/multiclass-evaluation-measures.html

[37] “Anchor Boxes for Object Detection - MATLAB & Simulink.” [Online]. Available : https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html

[38] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD : Single Shot MultiBox Detector,” *arXiv :1512.02325 [cs]*, vol. 9905, pp. 21–37, 2016, arXiv : 1512.02325. [Online]. Available : http://arxiv.org/abs/1512.02325

[39] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once : Unified, Real-Time Object Detection,” 2015, number : arXiv :1506.02640 arXiv :1506.02640 [cs]. [Online]. Available : http://arxiv.org/abs/1506.02640

[40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” *arXiv :1708.02002 [cs]*, Feb. 2018, arXiv : 1708.02002 version : 2. [Online]. Available : http://arxiv.org/abs/1708.02002

[41] Z. Tian, C. Shen, H. Chen, and T. He, “FCOS : Fully Convolutional One-Stage Object Detection,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South) : IEEE, Oct. 2019, pp. 9626–9635. [Online]. Available : https://ieeexplore.ieee.org/document/9010746/

[42] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI : IEEE, Jul. 2017, pp. 936–944. [Online]. Available : http://ieeexplore.ieee.org/document/8099589/

[43] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, “A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit,” *Electronics*, vol. 10, no. 3, p. 279, Jan. 2021. [Online]. Available : https://www.mdpi.com/2079-9292/10/3/279

[44] “XNO-6120R/FNP.” [Online]. Available : https://admin.hanwha-security.eu/business-security-products/xno-6120rfnp/

[45] “AXIS Q1615-E Mk II Network Camera - Assistance produit | Axis Communications.” [Online]. Available : https://www.axis.com/fr-fr/products/axis-q1615-e-mk-ii/support

[46] “OpenCV.” [Online]. Available : https://opencv.org/

[47] M. Genet, “Détection automatique des visages et plaques d’immatriculation dans des scènes routières,” Cerema, Université de Strasbourg, Tech. Rep., 2019.

[48] “Keras RetinaNet,” Aug. 2022, original-date : 2017-08-14T09 :14 :29Z. [Online]. Available : https://github.com/fizyr/keras-retinanet

8 Annexe

Configuration de la caméra extérieure

Cadrages de la caméra Tout au long du projet, le cadrage et la résolution des vidéos de la caméra extérieure ont été modifiés. Il y a pour l'instant trois cadrages différents, visibles en fig. 48. Dans l'ordre chronologique :

- **1280x720-a**, troisième en partant de la gauche.
  Utilisé du 10/10/2021 au 12/01/2022
- **1280x720-b**, premier en partant de la gauche.
  Utilisé du 13/01/2022 au 07/02/2022
- **800x600**, deuxième en partant de la gauche.
  Utilisé à partir du 08/02/2022

Figure 48 – Trois cadrages différents

Contrairement aux deux premiers, le cadrage **800x600** a un ratio $4:3$ qui déforme les images. Pour retrouver les bonnes proportions, il est nécessaire de changer les proportions, ce qui peut se faire par un redimensionnement en 950x600.

Pour référence, voici les tailles exactes ainsi que les décalages utilisés pour faire correspondre ces trois cadrages sur la figure 48 :

- **1280x720-b** : taille 1280x720, position relative $(0, 0)$ (référence)
- **800x600** : taille 939x594, position relative $(96, 69)$
- **1280x720-a** : taille 1694x896, position relative $(276, 60)$

Images par seconde dans les vidéos La fréquence d'enregistrement des images n'est pas non plus la même pour tous les fichiers vidéo. Voici un bref historique des différentes

fréquences d'images utilisées :

- du 20/10/2021 au 07/02/2022 : 3 im/s
- du 08/02/2022 au 21/02/2022 : 30 im/s
- du 22/02/2022 au 29/04/2022 : 20 im/s
- du 30/04/2022 au 02/05/2022 : pas d'images
- le 03/05/2022 : 10 im/s
  Cas particulier : 7 im/s jusqu'au fichier `027_20220503_115128.avi`
- Du 04/05/2022 au 16/05/2022 : 10 im/s
- À partir du 17/05/2022 : 6 im/s

Fichier journal pour le détramage des vidéos


```
    "152_20220401_132028": {
      "nombre images extraites": 43,
      "Images": {
         "1": 6,
         "2": 12,
         "3": 18,
         "4": 24,
...
         "40": 240,
         "41": 246,
         "42": 252,
         "43": 258,
     }
```


Liste des journées réservées à la base de test (pour la classification binaire)

- Journée du 29/10/2021
- Journée du 02/11/2021
- Journée du 03/11/2021
- Journée du 04/11/2021
- Journée du 26/11/2021
- Journée du 07/12/2021
- Journée du 26/12/2021
- Journée du 27/12/2021
- Journée du 13/01/2022
- Journée du 06/02/2022

Annotation dans LabelImg

Le tableau table 7 répertorie les labels utilisés pour l'annotation dans LabelImg.

Table 7 – Labels utilisé pour l'annotation dans LabelImg

Label	Description	Image
adr_symb_vide	*Emplacement plaque étiquette vide (pas annoté pour l'instant)*	
adr_symb_nde	Matière dangereuse pour l'environnement	
adr_symb_2-1_or_3	Gaz inflammables ou liquides inflammables	
adr_symb_md27	Produits chauds	
adr_symb_2-2	Gaz non inflammables, non toxiques	
adr_symb_5-1	Matières comburantes	
adr_symb_8	Matières corrosives	
adr_symb_9	Matières et objets dangereux divers	
adr_code_vide	*Emplacement panneau orange vide (pas annoté pour l'instant)*	
adr_code_orange	Panneau orange vierge, utilisé pour le transport de matières multiples ou sans numéro ONU spécifique	
adr_code_###_####	Panneau orange avec un code matière : 22_2187, 23_1965,30_1202, 33_1203, 80_1824, 80_1830, 90_3082, 99_3257, 223_1972 ... Exemple : adr_code_33_1203	
adr_code_unknown	Panneau orange illisible