LES JEUNES IHEDN

[RECHERCHE]

LA VISION PAR ORDINATEUR

UNE APPLICATION CLEF DE L'IA POUR LA DÉFENSE

Par Thomas E. | Groupe d'études scientifiques et techniques

Ce texte n'engage que la responsabilité de l'auteur. Les idées ou opinions émises ne peuvent en aucun cas être considérées comme l'expression d'une position officielle de l'association Les Jeunes IHEDN.

À PROPOS DE L'ARTICLE

Les modèles de vision par ordinateur présentent de multiples avantages opérationnels pour les applications de défense : précision du ciblage, navigation autonome, raccourcissement des boucles de commandement ou encore optimisation de ressources RH. Au-delà des méthodes mathématiques et statistiques classiques, les chercheurs développent depuis les années 1990 des approches *deep learning* qui ont révolutionné le domaine par leurs performances inouïes. En particulier depuis l'explosion de l'IA des années 2010-2020, ces nouveaux modèles de la vision se sont généralisés. Afin de comprendre leurs architectures, leurs tenants et aboutissants technologiques et les défis actuels du domaine, il est nécessaire de présenter un certain nombre de concepts techniques fondamentaux. Il sera ainsi possible de construire et comprendre des exemples, illustrations et réflexions opérationnelles.

Écrit dans le cadre des activités du Groupe d'études scientifiques et techniques, cet article propose donc de monter d'un cran dans la vulgarisation technologique. S'il a été écrit de manière à partir des notions de base et d'avancer progressivement, il semble clair qu'il peut nécessiter quelques prérequis (relativement simples) en matière d'ingénierie afin de le lire facilement. Il s'adresse donc à tous, mais n'intéressera peut-être pas tout le monde.

À PROPOS DE L'AUTEUR

Thomas E. est ingénieur. Spécialisé notamment en intelligence artificielle, il a travaillé sur sur plusieurs applications de l'IA dans des entreprises de la défense et pour le ministère des Armées.

SOMMAIRE

Introduction

La vision par ordinateur est un domaine absolument majeur des sciences appliquées et de l'intelligence artificielle. En un mot, il s'agit de concevoir des techniques permettant d'automatiser, dans un ordinateur, la compréhension plus ou moins haut niveau que le système visuel humain peut obtenir des images qu'il observe. Concrètement, on s'intéresse à extraire de ces dernières des caractéristiques sémantiques, texturales ou géométriques, afin d'obtenir des informations pour la décrire, la traiter et l'analyser.

Les applications en sont nombreuses et relativement directes : pour détecter des entités dans des images, pour « segmenter » l'image en zones sémantiques cohérentes, pour identifier des visages ou encore pour reconnaître automatiquement du texte sur des photos… On retrouve aujourd'hui ces algorithmes dans tous les domaines. Par exemple pour l'imagerie médicale, dans les véhicules autonomes, pour la sécurité civile, et bien évidemment pour la défense.

Il est alors possible d**'identifier avec précision des cibles** – bâtiments, drones, personnels, etc. –, de **détecter rapidement des comportements** suspects ou des changements sur des vidéos, d'**automatiser une partie du renseignement**, voire d'**améliorer l'autonomie des véhicules** ou la robustesse du **guidage** des missiles. Ces systèmes ne sont performants qu'à condition d'être correctement entraînés (données et hyperparamétrage de qualité), mais permettent alors un vrai avantage opérationnel sur les systèmes standard – seulement, toutefois, si les risques en sont acceptés.

La vision par ordinateur est donc devenue une composante incontournable d'un grand nombre de systèmes de défense modernes. Cet article propose de monter d'un cran dans la technicité de la vulgarisation habituellement proposée par les Jeunes IHEDN. L'idée n'est pas de réécrire un cours sur les systèmes de vision, mais plutôt de **donner assez de profondeur technologique pour comprendre les grandes composantes des algorithmes standards en la matière**. Cela permettra d'amener des exemples opérationnels et des réflexions sur les défis à venir et les choix doctrinaux.

I. Motivations

Avant d'entrer dans le cœur du sujet, la motivation présentée en avant-propos doit être complétée afin de comprendre **pourquoi l'IA a été intégrée dans les algorithmes de vision**, et pourquoi elle est devenue nécessaire pour avoir des systèmes performants aujourd'hui.

I.A. Petit historique technologique

La vision par ordinateur[1] est en réalité un domaine très ancien : les premières recherches datent des années 1960. À l'époque, les scientifiques et ingénieurs pratiquaient déjà **le traitement d'image**, qui est une autre approche d'analyse automatique d'images, **basée sur un paradigme algorithmique classique** (sans apprentissage machine) et des techniques mathématiques appliquées. Or, cette approche nécessite de paramétrer manuellement des filtres, des seuils et des descripteurs géométriques : elle atteint donc une limite lorsque les scènes deviennent trop complexes. Comment traiter par exemple des variations d'éclairage, de bruit capteur, d'échelles, d'occlusions … ou autres perturbations arbitraires ? Une solution consiste à adopter **une vision statistique et probabiliste**, qui supprime la nécessité des réglages fins manuels. C'est l'idée générale de ce que l'on appelle le *« machine learning »*, qui émerge dès les années 1980, y compris pour traiter les images. Ce nouveau paradigme permet à la machine « d'apprendre » plus ou moins automatiquement des descripteurs, c'est-à-dire d'optimiser elle-même ses réglages afin d'intégrer au mieux les données qui lui sont présentées dans un objectif de restitution donné. **Les premiers algorithmes de vision de ce type sortent dans les années 1990-2000** (exemple : pour traiter MNIST[2]), mais la vraie **rupture intervient dans les années 2010** – comme pour quasiment tous les autres systèmes d'IA. L'explosion des données, de la puissance de calcul et de la recherche en matière de réseaux neuronaux rend alors la vision par ordinateur vraiment fonctionnelle. Depuis, elle est sans cesse améliorée.

I.B. Traiter les masses de données

La première composante de la rupture de 2010 est la plus fondamentale : c'est l'explosion de la donnée disponible. Les raisons sont bien connues :

[1] « Vision par ordinateur ». *Wikipédia* [en ligne], 10/12/25 [consulté le 18/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Vision_par_ordinateur.

[2] « MNIST database ». *Wikipédia* [en ligne], 16/12/25 [consulté le 18/12/25]. Disponible sur : https://en.wikipedia.org/wiki/MNIST_database#MNIST.

démocratisation du numérique et des systèmes embarqués (ordinateurs et téléphones portables, voitures et montres connectées, etc.), miniaturisation des capteurs, progression des capacités de stockage et de transfert, etc. Très simplement, plus de systèmes avec plus de capteurs qui échangent plus de données. Une estimation[3] prédit 181 zettaoctets (181 trilliards d'octets, c'est-à-dire 181 000 milliards de Go) de données créées en 2025, contre 2 zettaoctets en 2010, soit une multiplication par 90 en 15 ans.

Cette démarche de « numérisation du monde » est une opportunité comme un défi, étant donné que la donnée brute reste inexploitable. En particulier, l'intérêt des opérateurs des systèmes de défense n'est pas nécessairement d'accumuler le maximum de données (bien que…), mais surtout de savoir en tirer les informations recherchées. Ainsi, un agent de renseignement voudra détecter un visage précis dans une foule, un planificateur d'opérations voudra compter le nombre de bateaux de la flotte ennemie imagée par un satellite, un industriel voudra que son missile reconnaisse sa cible, et ainsi de suite.

L'enjeu est donc de savoir extraire le signal pertinent au bon moment à l'intérieur des données : c'est relativement facile lorsque ces-dernières sont limitées en volume, mais comment faire lorsqu'elles s'accumulent en une masse et avec une vitesse telles qu'aucun opérateur humain ne peut les traiter ? Pour cette raison, on cherche à automatiser.

Clarifions un point clef : des études estiment certes que le cerveau humain traite une image plus rapidement qu'un ordinateur (13 ms (MIT[4]) contre 20 à 100 ms), mais ce traitement atteint des niveaux de précision, de robustesse et de fiabilité extrêmes. Et ce n'est pas forcément un avantage ! Le cerveau est contraint biologiquement en nombre de flux parallèles (3 ou 4 maximum), et son attention diminue avec l'accumulation de la fatigue. **Or les ordinateurs** – certes un peu plus lents – **traitent plusieurs centaines de flux simultanément, sans fatigue, à latence constante et à niveau de précision fixé** au nécessaire. Ainsi, si le cerveau est encore imbattable en performance ponctuelle, il ne peut gérer la masse, la simultanéité et la tenue dans la durée, aussi bien que la machine.

I.C. Avantages opérationnels de l'IA pour la défense

En matière militaire, il est pourtant clair que l'on ne peut se permettre ni latence excessive, ni attention diminuée au cours du temps. Il faut que les systèmes

[3] Direction des Affaires Juridiques. *Explosion des données numériques* [en ligne]. 09/02/2023 [consulté le 18/12/25]. Disponible sur : https://www.economie.gouv.fr/daj/lettre-de-la-daj-explosion-des-donnees-numeriques.
[4] TRAFTON, Anne. « In the blink of an eye ». *MIT News Office* [en ligne]. 16/01/24 [consulté le 18/12/25]. Disponible sur : https://news.mit.edu/2014/in-the-blink-of-an-eye-0116.

puissent être à tout moment au maximum de leur performance. C'est ici que l'IA entre en jeu : si le traitement d'images classique permettait déjà d'exploiter les avantages de la machine sur le cerveau, **l'IA démultiplie ce potentiel en réduisant le temps de traitement unitaire et en améliorant la performance**. Mettre des nombres sans présenter les métriques – cela sera fait plus tard – a peu de sens, mais on peut donner l'ordre de grandeur suivant : là où une technique standard d'aujourd'hui réussit sa tâche dans 80 ou 85 % des cas, un algorithme de *deep learning* peut approcher les 99 %. Sans parler de la complexité des tâches qu'il peut exclusivement résoudre. L'approche standard reste pertinente dans des cas simples ou contraints, mais les cas militaires généraux ne le sont pas *a priori* : par exemple, on y rencontre des flux vidéo multiples, des objets très complexes, des plages de longueurs d'ondes rares, du bruit capteur parfois extrême, des masses de données colossales, des leurres visuels très spécifiques, etc.

Par ailleurs, la vision par ordinateur peut être embarquée dans des systèmes plus ou moins autonomes : au-delà de la simple amélioration des performances, cela ouvre un tout nouveau pan d'applications. Je donne ici trois exemples.

L'exemple dont on entend peut-être le plus parler est celui des drones aériens. D'un côté, il y a le volet « lutte anti-drones », dont les algorithmes de vision permettent d'améliorer la précision pour repérer, cibler, suivre et éventuellement engager des drones intrus. L'autre volet concerne les systèmes embarqués : l'IA, rendue suffisamment frugale, peut être exécutée directement sur le drone en vol pour lui permettre de mieux naviguer (éviter les obstacles, se repérer en altitude, suivre des repères géographiques, gérer les situations *« GNSS-denied* ») et de mieux verrouiller sa cible[5] (détection et suivi temporel de la cible à partir d'un flux vidéo, reconnaissance d'obstacles et prise de décision en temps réel, classification entre éléments hostiles à engager et éléments pacifiques à éviter, etc.). C'est aussi un cas d'usage intéressant pour les missiles anti-aériens[6] : embarquer une caméra et un tel algorithme constitue une excellente contre-mesure aux leurres thermiques largués par les avions. Le missile amélioré, au-delà de simplement verrouiller le point le plus chaud, est désormais capable de suivre visuellement la forme de l'avion ciblé. En fusionnant cette information avec le suivi thermique, il consolide son guidage et le rend insensible au leurrage classique. Si certains de ces usages étaient déjà permis par l'approche standard, l'IA en améliore la performance et permet de nouvelles fonctionnalités.

[5] KUSHNIKOV, Vadim. « Drones with 'machine vision' are being mass-produced in Ukraine ». *Militarnyi* [en ligne]. 16/05/24 [consulté le 18/12/25]. Disponible sur : https://militarnyi.com/en/news/drones-with-machine-vision-are-being-mass-produced-in-ukraine/.

[6] Aviation. *Image recognition in air to air missiles* [en ligne]. 18/05/24 [consulté le 18/12/25]. Disponible sur : https://aviation.stackexchange.com/questions/105259/image-recognition-in-air-to-air-missiles.

En matière de drones, **un autre exemple est la navigation autonome** – on parle ici plutôt de drones terrestres ou maritimes. À l'instar d'une voiture autonome[7], le drone devient capable de « segmenter » l'environnement qu'il observe : où est la route ? Où sont les fossés ? Où sont les autres véhicules ? Et ainsi de suite. De cette manière, les algorithmes de navigation (par ailleurs fondés ou non sur l'IA – c'est un autre sujet) peuvent inclure ces informations en temps réel pour enrichir leur compréhension de leur environnement immédiat, et éventuellement adapter la trajectoire. En effet, certaines contre-mesures relativement simples ne peuvent que difficilement être prises en compte par un système entièrement autonome non doté de vision : une mine antichar, une tranchée récemment creusée ou un drone intercepteur nécessiteront probablement l'intervention d'un humain pour être complètement comprises et évitées… sauf si le système embarqué est capable de le faire. Cela pose un enjeu de doctrine, développé plus tard. Mais sans s'en soucier pour le moment, on comprend donc en quoi embarquer une capacité de vision améliore l'autonomie des systèmes.

Un dernier cas d'usage typique est **l'analyse d'images pour le renseignement militaire**. Par exemple, depuis les années 1960-1970, l'imagerie par satellites est devenue centrale[8] dans toutes les préparations stratégiques et opérationnelles : elle permet en effet d'espionner tous les territoires ennemis et de photographier l'état de ses forces militaires exposées. Cependant, compte tenu de l'augmentation constante du nombre de satellites en orbite et de l'amélioration des taux de revisites (c'est-à-dire de l'augmentation de la fréquence de rafraîchissement des images, une des raisons en étant le recours accru aux constellations de satellites), les masses d'images à analyser deviennent trop importantes, et leur qualité croissante augmente démentiellement la taille. Par ailleurs, les zones de conflit et la distribution géographique des forces se multiplient à une vitesse telle que les analystes ne peuvent pas tout suivre. S'il était encore possible de compter manuellement les véhicules ennemis sur les images satellites en 1991[9], il est nécessaire aujourd'hui d'automatiser cette mission pour recentrer les analystes sur de « vraies » tâches d'analyse. Là encore, l'IA le permet avec une précision inégalée, un traitement parallèle optimisé et une attention constante.

En somme : les algorithmes de vision par ordinateur sont devenus indispensables dans beaucoup de systèmes de défense. Le paradigme

[7] COHEN, Jeremy. « L'IA … et le véhicule fut autonome ! ». *Medium* [en ligne]. 24/01/18 [consulté le 18/12/25]. Disponible sur : https://medium.com/france-school-of-ai/lia-et-le-v%C3%A9hicule-fut-autonome-2abd8a348eb5.

[8] BARRE, Joël. « Des satellites pour gagner la guerre de l'information ». *La Jaune et la Rouge* [en ligne], décembre 2001. Disponible sur : https://www.lajauneetlarouge.com/wp-content/uploads/2013/01/570-page-027-030.pdf?srsltid=AfmBOop_JN3pk4CPn-WDWDO357azDn5LxF1BLgl4o1zcT0bLy-ouQwMe.

[9] GROUARD, Serge. « Le rôle des satellites américains dans la région du Golfe ». *Revue Défense Nationale* [en ligne], décembre 1990n n°515. Disponible sur : https://www.defnat.com/e-RDN/vue-article.php?carticle=6199&cidrevue=515.

traditionnel fonctionne encore dans des cas contraints et bien connus, mais les avancées en matière d'IA, bien plus flexibles, tendent à les remplacer et ont permis de déployer les systèmes de vision à beaucoup **plus large échelle**. Elle ouvre de **nouvelles applications**, notamment vis-à-vis de l'autonomie des systèmes, et permet de **traiter avec une efficacité inouïe les masses colossales de données** qui caractérisent tant les champs de bataille que la préparation des opérations.

II. Introduction technique et concepts fondamentaux

Plongeons désormais dans la réalisation technique ; encore une fois, l'objectif n'est pas de copier un cours, mais de tenter d'introduire les concepts fondamentaux de la vision par ordinateur, qui permettront de comprendre les tenants et aboutissants de ce type de technologie.

II.A. Les réseaux de neurones[10] [11] pour les nuls

Il y a des réseaux de neurones dans notre cerveau : ils sont très complexes, mais c'est en s'en inspirant que les premiers chercheurs, dans les années 1950, parvinrent à construire le premier « neurone formel »[12]. Concrètement, il s'agit d'**un modèle mathématique très simple qui permet d'implémenter des fonctions** logiques ou arithmétiques. C'est une « machine » qui transforme des signaux d'entrée en signaux de sortie, selon une règle mathématique prédéfinie, en 2 temps. D'abord, une combinaison linéaire : multiplier les signaux d'entrée par des coefficients respectifs donnés (dits « poids ») ; à laquelle ajouter un nombre donné (dit « biais »). Ensuite, faire passer le tout dans une fonction donnée (dite « d'activation »).

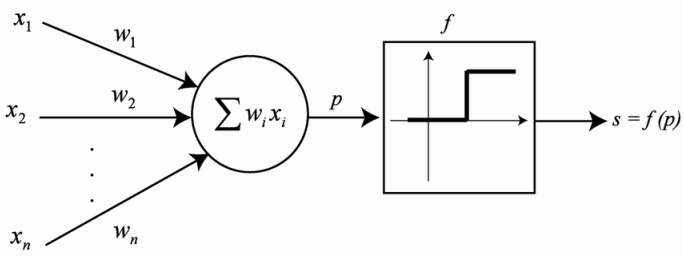


Schéma d'un neurone formel. Source : ResearchGate

Par la suite, afin de complexifier les opérations possibles, **ces mêmes chercheurs ont l'idée naturelle d'assembler de tels neurones entre eux, couche après couche**, comme dans l'idée que l'on se fait du cerveau. **C'est tout simplement cet assemblage que l'on appelle un « réseau de neurones ».** Toutefois, les paramètres de chacun de ces neurones (poids, biais) sont pour l'instant fixés, et les fonctions d'activation – choisies non-linéaires pour introduire de la complexité dans le raisonnement simulé – sont encore relativement rudimentaires. On comprend

[10] Wikipédia. *Réseau de neurones artificiels* [en ligne]. 17/12/25 [consulté le 19/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels.
[11] LEE, Fangfang. « Que sont les réseaux de neurones ? ». *IBM* [en ligne]. Consulté le 19/12/25. Disponible sur : https://www.ibm.com/fr-fr/think/topics/neural-networks.
[12] MCCULLOCH, Warren & PITTS, Walter. « A logical calculus of the ideas immanent in nervous activity ». *Bulletin of Mathematical Biology* [en ligne], 1990. Vol. 52 No. 1/2, pages 99-115. Disponible sur : https://acrobat.adobe.com/id/urn:aaid:sc:EU:3c7b75d3-d749-4fc5-a386-b8baf7e474fe.

bien qu'une limite va rapidement apparaître. Plus le nombre de neurones et de couches augmente, et plus il est difficile de régler manuellement tous ces paramètres : d'abord parce qu'il y en a trop ; ensuite parce qu'on perd totalement le sens de ce que chaque paramètre signifie ; enfin parce qu'un réglage manuel ne garantirait pas l'optimalité du résultat attendu.

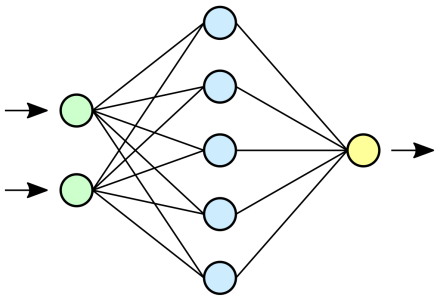
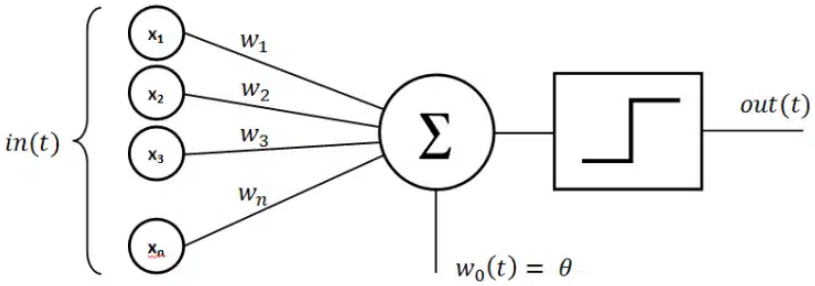


Schéma d'un réseau de neurones simple. Source : Wikipédia

La solution a été proposée par un autre chercheur, Frank Rosenblatt, quelques années plus tard : **faire apprendre automatiquement au réseau ses propres paramètres**[13]. L'architecture du réseau est extrêmement basique, mais introduit une nouveauté : un mécanisme d'apprentissage supervisé permettant d'ajuster les poids automatiquement en fonction de l'erreur commise par le réseau. En pratique, **le caractère supervisé signifie que le chercheur utilise des ensembles {entrées données ; sortie de référence pour ces entrées} connus afin d'évaluer le réseau pendant son apprentissage**.

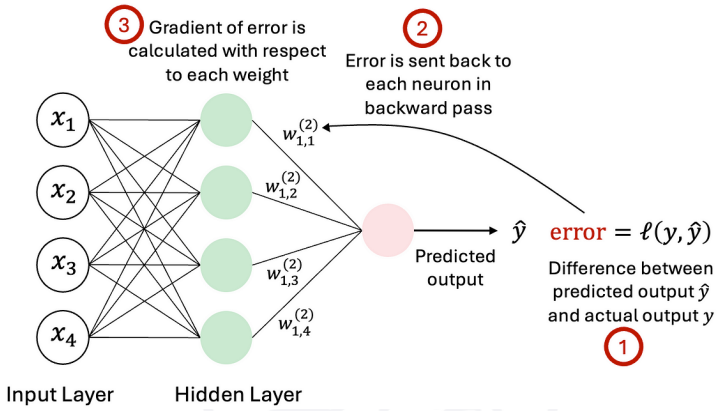


Un perceptron. Source : DataScientest

Pour des entrées données, on « propage » l'information dans les couches du réseau jusqu'à la sortie : concrètement, on fait les calculs à l'intérieur de chaque neurone pour chaque couche, puis on passe à la suivante, jusqu'à obtenir une sortie. **Cette dernière peut être comparée à la sortie de référence – connue, donc –, ce qui**

[13] ROSENBLATT, Franck. « The perceptron: a probabilistic model for information storage and organization in the brain ». *Psychological Review* [en ligne], 1990. Vol. 65, n°6, 1958. Disponible sur : https://www.ling.upenn.edu/courses/cogs501/Rosenblatt1958.pdf.

permet de mesurer une erreur. Si le réseau est bien réglé, cette erreur est petite et négligeable (pas forcément nulle) ; s'il est mal réglé, cette erreur est grande. **Grâce au calcul différentiel** (un outil mathématique, le « gradient », qui mesure comment la variation d'une quantité entraîne la variation d'une autre)**, on peut propager cette erreur « en arrière » dans le réseau** (*« backpropagation »*) **afin d'ajuster automatiquement les poids** de chaque neurone en fonction de leur influence respective sur la sortie. Concrètement, on « remonte » le réseau de la sortie vers l'entrée, en faisant les calculs de gradients « à l'envers » (puisque le gradient original ne peut être calculé qu'à la sortie). En répétant cette opération de propagation / *« backpropagation »* un grand nombre de fois, on ajuste petit à petit les poids vers des valeurs optimales sur l'ensemble {entrées données ; sortie de référence pour ces entrées}. C'est ainsi que le réseau « apprend ».



Le principe de la backpropagation*. Source :* Medium

$$\frac{d}{dx}\left[\left(f(x)\right)^n\right] = n\left(f(x)\right)^{n-1} \cdot f'(x)$$

$$\frac{d}{dx}\left[f\left(g(x)\right)\right] = f'\left(g(x)\right)g'(x)$$

Le principe du théorème de dérivation des fonctions composées (« chain rule »*),*

astuce clef pour rétropropager le gradient lors de l'entraînement. Source : Calcworkshop

Il y a un certain nombre d'astuces techniques permettant de faire en sorte que tout cela fonctionne. Par exemple :

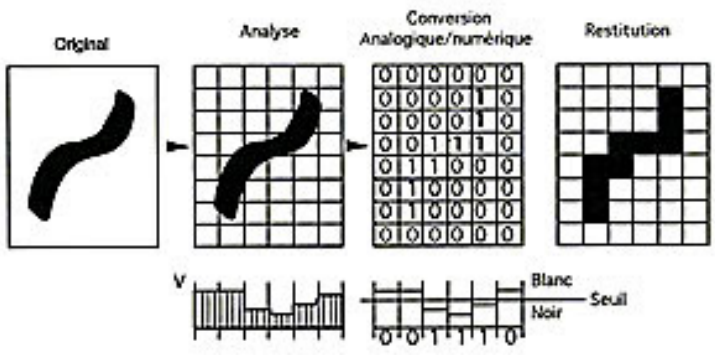
- Comment s'assurer que le réseau est capable d'atteindre la bonne complexité pour apprendre la fonction sous-jacente aux données qu'on lui montre (choix des **fonctions d'activation** et de l'**architecture** du réseau : varier le nombre de neurones par couches ou introduire des récurrences par exemple) ?

- Comment optimiser l'évaluation de l'erreur en sortie (choix d'une **« fonction de perte »**, appliquée à la différence sortie effective / sortie de référence) ?
- Comment éviter que le réseau n'apprenne par cœur toutes les données d'entraînement, sans vraiment « comprendre », c'est-à-dire être capable de généraliser (problématique d'***overfitting***) ?
- Comment ajuster la qualité, la vitesse, voire la structure de l'apprentissage (choix dits « d'**hyperparamètres** », choix des **pré-traitement** des données, choix de la division en « passes » d'apprentissages) ?

Encore une fois, l'objectif ici n'est pas d'entrer dans tous ces détails techniques. Concluons donc simplement cette phase d'introduction aux réseaux de neurones par un superbe théorème qui leur donne tout leur sens : **le théorème d'approximation universelle**. Il en existe plusieurs raffinements successifs, mais la 1ère version date de 1989[14]. L'idée est simplement de dire qu'un réseau suffisamment large, muni d'une fonction d'activation non linéaire, peut approcher arbitrairement bien toute fonction continue sur un domaine borné. Ce théorème a beaucoup de limites : il ne dit rien ni sur la vitesse d'apprentissage, ni sur le nombre de neurones nécessaires, ni sur les fonctions discontinues ou sur la généralisation. **En revanche, il formalise l'intuition selon laquelle « les réseaux de neurones fonctionnent »** pour approximer n'importe quelle fonction mathématique, et donc n'importe quelle tâche de vision associée**.**

II.B. Notions clefs de la vision par ordinateur

Pour pouvoir appliquer ces réseaux de neurones à la « compréhension » d'images, il faut d'abord comprendre ce qu'est une image numérique. Cela invite à présenter des notions basiques en matière de traitement d'image traditionnel, qui seront utiles par la suite.

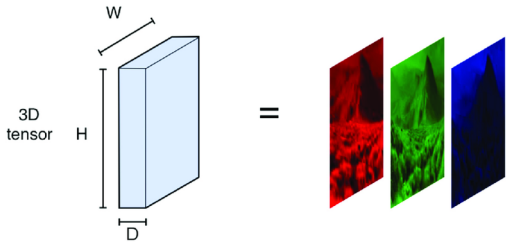


Restitution numérique d'une image analogique. Source : www.map.toulouse.archi.fr

[14] « Théorème d'approximation universelle ». *Wikipédia* [en ligne], 09/05/25 [consulté le 19/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_d%27approximation_universelle.

Une image numérique est simplement une structure de données[15], c'est-à-dire une convention informatique selon laquelle des informations sont enregistrées en mémoire. Le choix de cette structure pour stocker une image conditionne la manière dont les algorithmes, ensuite, pourront analyser et interpréter le contenu visuel. Chaque choix de structure implique des avantages (meilleure définition, analyse et manipulation facilitées, etc.) et des inconvénients (taille au stockage, manipulation complexe, etc.). Il existe deux grands types de structures pour stocker les images : le mode vectoriel (utiliser des formules géométriques pour décrire les éléments visuels) et **le mode matriciel** (qui utilise donc un tableau de pixels, « *picture elements* »). Seul ce second type sera évoqué, car il est beaucoup plus fondamental et est globalement le seul utilisé dans les algorithmes de vision.

En mode matriciel, on décompose l'image comme un tableau de pixels, c'est-à-dire en points élémentaires de l'image. Pour des raisons historiques et des limitations techniques (encodage sur 8 bits), chaque pixel est concrètement **encodé dans l'ordinateur comme un nombre entier**, souvent entre 0 (pour signifier l'absence de lumière) et 255 (pour signifier la lumière maximale) – ce choix de 256 valeurs possibles (de 1 à 255, plus le zéro). Il est ainsi possible d'encoder avec un seul nombre une valeur de luminance : ce nombre s'appelle **un « canal »**. Avec un seul canal, on ne fait donc qu'une image en noir et blanc. **Mais avec plusieurs canaux superposés, on peut encoder des décompositions plus riches :** typiquement, on utilise des canaux R (« *red* »), G (« *green* »), B (« *blue* »). Chaque pixel est alors décrit par 3 valeurs entre 0 et 255, ce qui permet d'encoder une gamme de couleurs suffisante. On peut rajouter des canaux pour d'autres bandes de fréquences ou d'autres nuances (transparence par exemple), mais le principe reste le même.



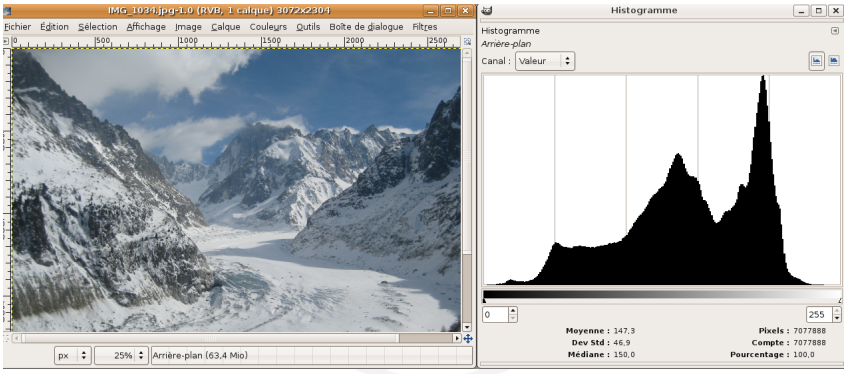
Superposition des canaux RGB. Source : ResearchGate

Donc une image en « langage ordinateur », c'est simplement une superposition de tableaux en deux dimensions. Chaque tableau représente un canal, et chaque case d'un tableau donné contient la valeur attribuée au pixel de l'image correspondant, pour le canal en question. Une image matricielle, telle qu'en prennent des capteurs embarqués et qu'en exploitent les algorithmes, c'est donc

[15] « Image numérique ». *Wikipédia* [en ligne], 04/11/25 [consulté le 19/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Image_num%C3%A9rique.

un empilement de [nombre de canaux] matrices de tailles [nombre de pixels en largeur ; nombre de pixels en hauteur] (en mathématiques, ça s'appelle un « tenseur » d'ordre 3).

À partir de cette représentation, il est possible de faire **des traitements statistiques**. Un outil fondamental du traitement d'image est par exemple **l'histogramme**, c'est-à-dire un graphique décrivant la distribution des intensités (valeurs) de pixels dans une image ou un canal donné. Concrètement : pour chaque valeur entière entre 0 et 255, on compte le nombre de pixels de l'image qui la prend. On obtient ainsi des informations de contraste, de luminance et de dominantes de couleur. **C'est une première analyse, mais ça n'informe en rien sur la structure spatiale de l'image, sur les objets y sont présents, sur les contours, etc.** Deux images très différentes peuvent présenter des histogrammes très proches (par exemple : une photo de neige à la montagne, très blanche avec un peu rochers gris ; et une photo d'une lettre écrite sur un papier blanc*, idem*).



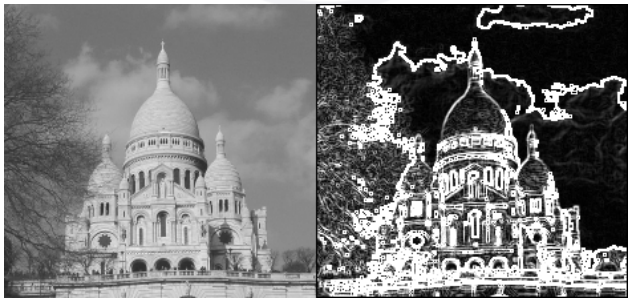
Une photographie et son histogramme. Source : Wikipédia

Pour aller au-delà, le traitement d'images traditionnel propose **la notion de filtres**[16]. Un filtre, c'est simplement une opération mathématique appliquée localement à chaque pixel, donc **dont le résultat est fonction de lui et de ses voisins immédiats**. Les filtres permettent des opérations un cran plus avancées que la simple analyse statistique : on peut ainsi lisser une image (réduire le bruit), faire ressortir les contours des objets, renforcer des textures, etc. Ce sont des opérations de base, sur lesquelles sont construites des méthodes plus avancées.

[16] Datacorner. *Traitement d'images (partie 6: FIltres & Convolution)* [en ligne], s.d. [consulté le 19/12/25]. Disponible sur : https://datacorner.fr/image-processing-6/.

Application d'un filtre médian pour débruiter une image. Source : OpenClassrooms

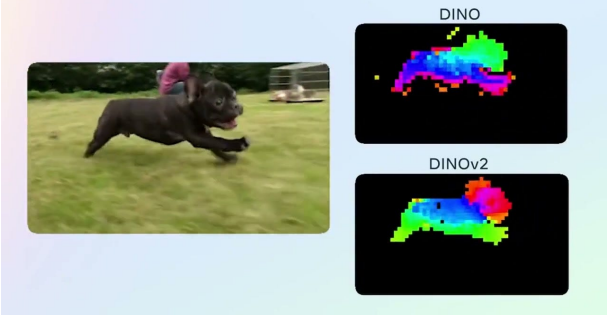
Une opération fondamentale qui nous intéresse dans le contexte de vision par ordinateur, c'est **la détection de motifs et, *a fortiori*, la reconnaissance d'objets**. Un exemple consiste à imager un missile qui poursuivrait un avion : l'objectif n'est pas de faire ressortir les contours des nuages, ni d'accentuer la couleur du ciel… mais bien détecter la forme de la cible pour pouvoir la verrouiller. Or, qu'est-ce qu'une cible ? C'est-à-dire, qu'est-ce qu'un objet ? **En vision, on considère que les objets sont représentés sur les images par des « caractéristiques » – *« features »* en anglais** (c'est le terme couramment utilisé en IA). Les caractéristiques peuvent être très diverses : des lignes, des coins, des formes spécifiques (carré, rond, disphénoïde adouci, peu importe), des textures particulières, des motifs de couleur, des unités sémantiques, et même (voire surtout) des concepts que l'humain ne comprend pas tels quels, étant donné que le mystère du *deep learning* est de faire découvrir puis apprendre à la machine ses propres caractéristiques qui lui permettent le mieux de « comprendre » les images sur sa tâche.



Application d'un filtre de Sobel pour détecter les contours dans une image. Source : Wikipédia

Avec des filtres simples ou des méthodes statistiques localisées, on peut déjà extraire des motifs basiques : voir ci-dessus un exemple de filtre de Sobel[17], qui extrait les contours des objets dans l'image en calculant les gradients des pixels alentour.

[17] « Filtre de Sobel ». *Wikipédia* [en ligne], 30/06/25 [consulté le 19/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Filtre_de_Sobel.



Cartes de features *obtenues par DINO et DINOv2. Source :* Meta

Dans cette continuité, on cherche donc à obtenir des caractéristiques plus complexes et en plus grand nombre, afin de pouvoir les combiner et transformer ainsi sans ambiguïté des images matricielles complexes en représentations structurées de caractéristiques – c'est-à-dire d'ensembles mutuellement exclusifs que l'ordinateur peut ainsi classifier, détecter ou segmenter. **Dans les approches traditionnelles, le choix des caractéristiques repose sur l'expertise humaine**, dans la mesure où il faut choisir à l'avance quelles caractéristiques sont importantes pour une tâche donnée. **Le *deep learning* lève cette contrainte : comme mentionné précédemment, il construit lui-même ses ensembles de caractéristiques.**

Toutefois, avant de plonger vraiment dans le *deep learning*, il reste une opération mathématique à présenter : **la convolution**[18]. C'est absolument fondamental dans tous les systèmes de vision par ordinateur (et dans le traitement du signal de manière générale). La notion de filtre a déjà été présentée ci-avant : en réalité, **la convolution consiste simplement à faire glisser un de ces filtres sur plusieurs pixels successifs**. À chaque pixel, on calcule une combinaison pondérée des pixels voisins, définie par le filtre choisi (appelé « noyau de convolution » ou *« kernel »*[19]). Cette opération permet de détecter des motifs spécifiques et plus complexes. Le filtre de Sobel, illustré ci-dessus, est le résultat de l'application d'un filtre (matrice) particulier dans un produit matriciel de convolution avec la matrice image.

[18] « Produit de convolution ». *Wikipédia* [en ligne], 02/10/25 [consulté le 19/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Produit_de_convolution.
[19] « Noyau (traitement d'images) ». *Wikipédia* [en ligne], [en ligne]. 06/12/25 [consulté le 19/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Noyau_(traitement_d%27image.

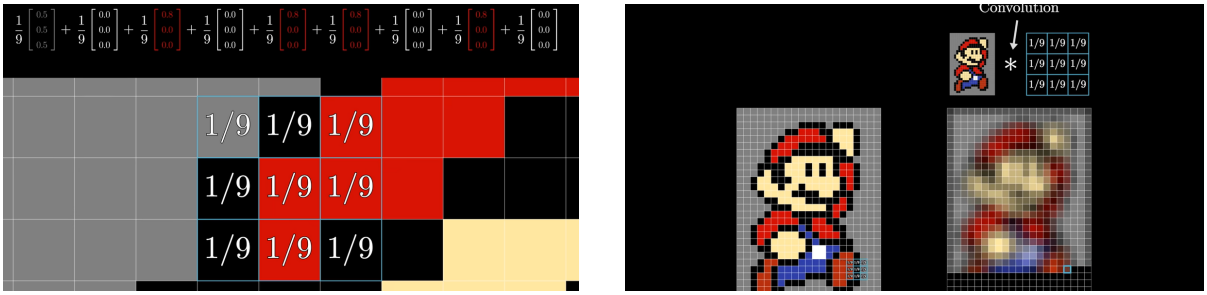


Schéma explicatif du produit de convolution. Source : 3Blue1Brown[20]

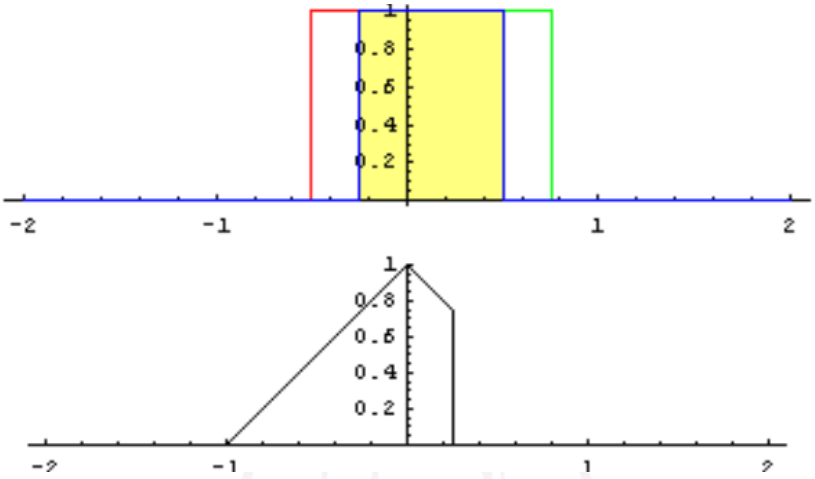


Schéma un peu plus théorique du produit de convolution. Il faut imaginer le signal vert se déplacer de la gauche vers la droite, créant donc une plus ou moins grande zone d'intersection (en jaune). Source : Wikipédia

En pratique, plusieurs paramètres contrôlent la manière dont la convolution est appliquée. On peut en mentionner deux principaux : **le pas de déplacement** (indiquant de combien de pixels le noyau choisi est décalé à chaque étape)**, et le « *padding* »** (indiquant la manière dont l'on traite les bords de l'image, par exemple en ajoutant des pixels fictifs d'une couleur arbitraire). Ces choix semblent très techniques mais ont un vrai impact sur la performance du traitement.

Schéma de padding. *Source :* GeeksforGeeks

[20] 3Blue1Brown. *But what is a convolution?* [vidéo en ligne]. 18/11/22 [consultée le 19/12/25]. Disponible sur : https://youtu.be/KuXjwB4LzSA?si=Ad1miheeV-6bC8By.

II.C. Réseaux convolutifs

Ces éléments de base sont nécessaires pour la suite : les capteurs prennent des images matricielles, qui sont analysées par des techniques de traitement afin d'en extraire des caractéristiques. Ces dernières peuvent être soit prédéfinies manuellement (approches traditionnelles), soit apprises automatiquement. Pour rappel, on a vu précédemment que les réseaux de neurones, bien conçus et bien entraînés, pouvaient approximer n'importe quelle fonction (sous conditions). Cette fonction peut être celle de l'extraction des meilleures caractéristiques pour réaliser une tâche donnée. On comprend donc pourquoi les réseaux de neurones sont utilisés pour les tâches de vision : meilleures caractéristiques, meilleure performance sur des tâches complexes, meilleure généralisation… Voyons comment ça fonctionne.

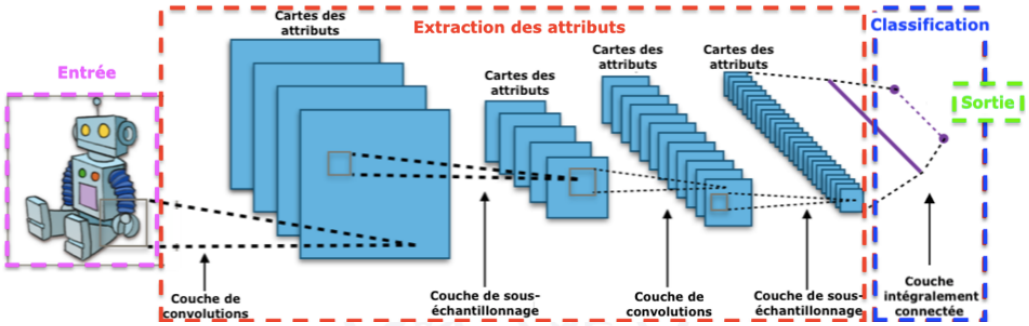


Schéma d'architecture CNN. Source : Wikipédia

Comme présenté ci-avant, la convolution permet d'exploiter la structure spatiale des images et de détecter des motifs locaux. Or, en parallèle, les réseaux de neurones permettent d'apprendre des fonctions automatiquement à partir d'un ensemble de données et d'une tâche donnée. De manière relativement intuitive, **des scientifiques ont donc eu l'idée d'intégrer les deux afin d'apprendre automatiquement les bons filtres afin de « faire comprendre » de manière optimale l'ensemble de données à la machine.** Un des premiers exemples est Kunihiko Fukushima, en 1980[21], qui présente un **« CNN » (*« Convolutional Neural Network »*)**, intégrant explicitement l'opération de convolution à l'intérieur d'un réseau de neurones. Là où les réseaux entièrement connectés traitent les entrées indépendamment, **les CNN tiennent compte de la forte corrélation entre pixels voisins**, ce qui permet un traitement plus intelligent et plus pertinent. En effet et par exemple, conserver un grand nombre de pixels bleus identiques n'est pas nécessaire pour décrire une zone de ciel. Il en va de même avec les motifs, dans la

[21] FUKUSHIMA, Kunihiko. « Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position ». *Biological Cybernetics* [en ligne], 1980, vol. 36. Disponible sur : https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf.

mesure où un trop grand nombre de pixels redondants n'est pas nécessaire pour les caractériser, à la condition de savoir le reconnaître… C'est ce que font les réseaux convolutifs[22] : réduire le nombre d'entrées à chaque couche, grâce à des opérations de convolutions apprises automatiquement, donc aptes à reconnaître les caractéristiques importantes du jeu de données.

Un CNN est typiquement implémenté en plusieurs étapes. **D'abord, des couches convolutives. Chacune est spécialisée dans des caractéristiques particulières, apprises automatiquement et de plus en plus complexes** (proche de l'entrée du réseau, on détecte des bords, des grandes formes, des orientations ; puis plus en profondeur, on détecte des motifs davantage sémantiques). **Ensuite, afin de réduire la taille des représentations intermédiaires** (c'est-à-dire des vecteurs de nombres traversant le réseau)**, les CNN intercalent des couches de sous-échantillonnage** entre les couches convolutives. On appelle cela le *« pooling »*, et cela permet donc de « réduire » ou « résumer » de l'information spatiale dans moins de nombres – tout en limitant par ailleurs l'*overfitting*. Une technique est le *« max pooling »* : parmi un petit nombre de pixels voisins, on ne garde que celui qui a la plus grande valeur. On peut faire pareil avec la moyenne, le minimum, ou des choses plus complexes… ça paraît un peu idiot, mais ça marche. En faisant se succéder des couches de convolution auto-apprises et des couches de *pooling*, **on arrive ainsi à construire des « représentations latentes » des images, c'est-à-dire des vecteurs de nombres représentant les caractéristiques localisées** (i.e. avec une information de localisation spatiale) permettant de décrire l'image. Il faut et il suffit ensuite d'ajouter et d'entraîner des couches spécifiques à la tâche, dans une dernière partie du réseau, afin d'extraire de ces représentations les informations souhaitées : où sont les objets d'intérêt ? Que représente telle ou telle partie de l'image ? Quelle classe attribuer à cette photo ? Etc. Ces couches-ci sont typiquement de type *« fully connected »*, c'est-à-dire très basiques, sans convolution, avec seulement des neurones interconnectés entre eux.

Un élément technique peut toutefois interroger : comment « fait-on entrer » une image dans un réseau de neurones ? On a vu précédemment qu'un neurone manipule des vecteurs de nombres, et qu'un réseau entièrement connecté attend donc en entrée un vecteur de grande dimension. Dans ce cadre, il est effectivement possible « d'écraser » une image matricielle – qui est un tenseur d'ordre 3 – en mettant bout à bout toutes ses valeurs dans un unique vecteur. Cette opération, appelée *« flatten »*, ne correspond à aucun calcul : elle consiste uniquement à réorganiser les données.

Cependant, cette approche devient rapidement impossible lorsque les images sont de grande taille, car le nombre de paramètres explose (donc les besoins en

[22] « Réseau neuronal convolutif ». *Wikipédia* [en ligne], 14/05/25 [consulté le 19/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/R%C3%A9seau_neuronal_convolutif.

mémoire vive). **Et puis surtout, cette approche détruirait totalement la structure spatiale de l'image, pourtant essentielle à l'analyse visuelle.** C'est précisément pour éviter ce problème que les réseaux convolutifs ont été introduits. Dans un CNN, l'image est fournie directement sous forme de tenseur à la première couche, et les opérations de convolution exploitent explicitement les relations locales entre pixels. L'aplatissement n'intervient que très tardivement, une fois que l'information a été progressivement compactée et abstraite. **L'enjeu fondamental est donc de préserver la logique spatiale de l'image aussi longtemps que possible :** si l'on mélange les pixels trop tôt, les couches de convolution perdent tout leur intérêt.

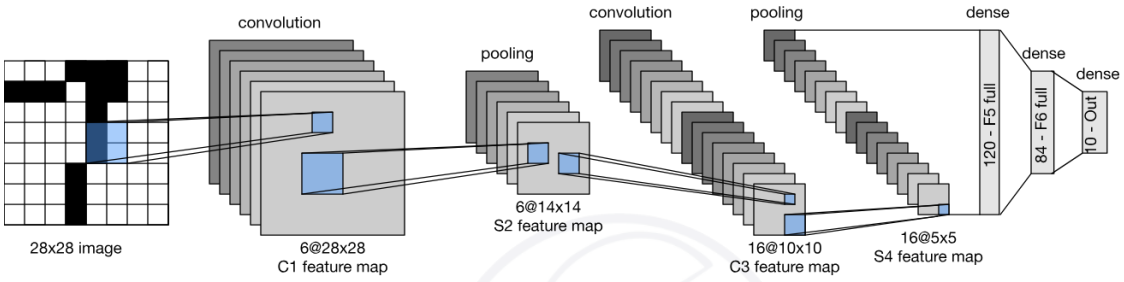


Schéma de LeNet. *Source :* Dive into Deep Learning

Les architectures convolutives sont la fondation des systèmes de vision par IA. Au fil du temps depuis 1980, plusieurs architectures emblématiques de réseaux convolutifs ont été proposées : voici une liste de celles qui semblent les principales[23].

- **LeNet**[24] **; fin des années 1990.** Proposé par Yann Le Cun (un français, parmi les pères fondateurs de l'IA au sens où on l'entend aujourd'hui) pour reconnaître des chiffres écrits à la main. C'était un *benchmark* typique des modèles de vision dans les années 90 : soit sur des jeux de données de chèques (ici, avec une précision de 99,3 %), soit sur le dataset MNIST, pour la poste américaine. **Ce fut le 1er CNN à être industrialisé** ;

[23] « Convolutional neural networks ». *ml4a.github* [en ligne], s.d. [Consulté le 19/12/2025]. Disponible sur : https://ml4a.github.io/ml4a/convnets/.

[24] LECUN, Yann ; BENGIO, Yoshua, HAFFNER ; Patrick, RACHMAD & Yoesoep Edhie. « Gradient-Based Learning Applied to Document Recognition ». *Proc. of the IEEE* [en ligne], novembre 1990. Disponible sur : https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition.

- **AlexNet**[25] **; 2012. C'est un exemple-type de la révolution des années 2010 :** énormes jeux de données, réseaux plus profonds et puissance de calcul très importante ont permis à AlexNet de remporter une compétition de classification d'images – en l'occurrence, sur le jeu de données ImageNet, comportant 14 millions d'images labellisées en 1000 classes. Les performances d'AlexNet surpassaient de loin toutes les autres solutions de l'époque ;
- **ResNet**[26] **; 2015. L'idée ici est d'implémenter des connexions « résiduelles »** (d'où le *« Res »*) entre couches du réseau non-successives, pour en court-circuiter certaines et permettre d'augmenter largement la profondeur du réseau (donc la complexité des représentations latentes profondes) tout en restant pertinents et en limitant certains problèmes techniques de *backpropagation ;*
- **U-Net**[27] **; 2015.** À l'origine développé pour la segmentation d'images médicales, **U-Net introduit une architecture dite « entièrement convolutive »**, reste une référence parmi les modèles ***« image-to-image »*** (transformation d'une image en entrée en une image en sortie). U-Net est constitué de 2 parties : une partie « contractante » (cycle descendant du U), et une « expansive » (partie ascendante), avec des connexions résiduelles entre les deux. Il n'y a donc pas de couche entièrement connectée. Cette architecture permet de traiter des images plus volumineuses et d'obtenir une meilleure performance lorsque la tâche nécessite de reconstruire une image en sortie.

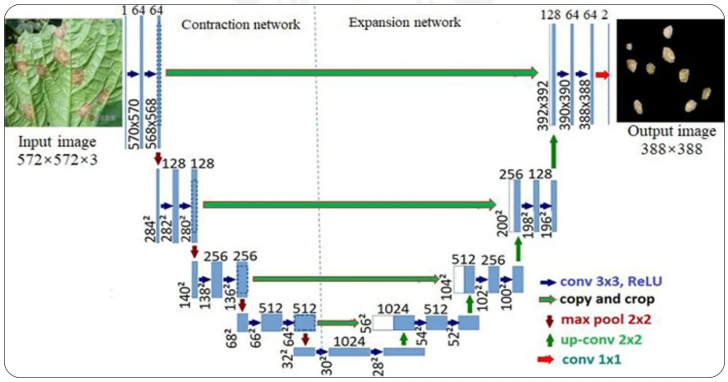


Schéma de U-Net*. Source :* Ultralytics

[25] KRIZHEVSKY, Alex ; SUTSKEVER, Ilya ; HINTON, Geoffrey E. « ImageNet Classification with Deep Convolutional Neural Networks ». *Proceedings.neurips* [en ligne], s.d. [Consulté le 19/12/25]. Disponible sur : https://proceedings.neurips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

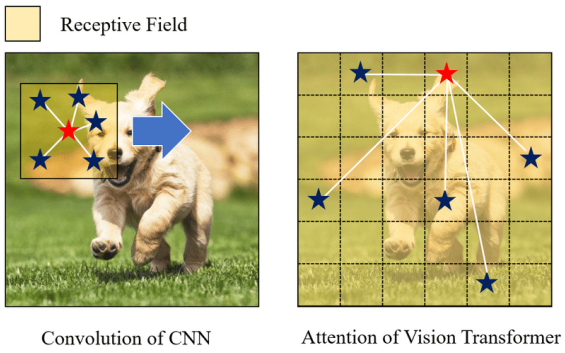
[26] HE, Kaiming ; ZHANG, Xiangyu ; REN, Shaoqing & SUN, Jian. « Deep Residual Learning for Image Recognition ». Arxiv [en ligne], 10/12/15 [Consulté le 19/12/25]. Disponible sur : https://arxiv.org/abs/1512.03385.

[27] RONNEBERGER, Olaf ; FISCHER, Philipp & BROX, Thomas. « U-Net: Convolutional Networks for Biomedical Image Segmentation ». Arxiv [en ligne], 18/05/15 [Consulté le 19/12/25]. Disponible sur : https://arxiv.org/abs/1505.04597.

II.D. Les *Transformers*

Ainsi, au fil des nouvelles idées d'intégration et d'architectures, les réseaux de neurones convolutifs ont connu plusieurs évolutions et améliorations de performances successives. **Toutefois, l'approche convolutive repose sur une hypothèse très forte : la localisation de l'information visuelle.** Concrètement, on suppose que pour comprendre une image dans son ensemble, il suffit de comprendre chacune de ses sous-parties locales. Cela fonctionne dans des cas simples : un bouquet de fleurs, c'est un ensemble de fleurs ; un avion dans le ciel, c'est du ciel partout plus un avion à un endroit ; un numéro sur un chèque, c'est un ensemble de chiffres ; etc. Cette hypothèse est valable dans beaucoup de cas, et explique la performance de l'approche CNN et la généralisation de son utilisation depuis les années 2010.

Toutefois, si l'on réfléchit à un cas plus général, il apparaît que cette hypothèse est fausse dans l'absolu. En effet, pour interpréter une image et aller au-delà de la simple observation de ses composantes, il est parfois nécessaire de s'intéresser aux dépendances à longue distance. Imaginons une photo d'un convoi de chars dans le désert. L'approche CNN n'aura aucun mal à détecter localement les véhicules : formes reconnaissables, contrastes de couleur avec le sol, éléments spécifiques (chenilles, antennes, armement, etc.). Pour autant, ce modèle en l'état sera-t-il capable d'interpréter l'image comme une photo d'un convoi organisé, se déplaçant dans une même direction, avec un *leader* et une structure de colonne éventuelle ? *A priori* non, car **les convolutions ne s'intéressent qu'aux localités et sont incapables de faire des liens à longue distance** : comparer les pixels du 1er char, par exemple en haut à droite, et celui du dernier char, et par exemple en bas à gauche, est hors de leur portée. Pour certaines tâches de classification, de sous-titrage ou de fusion de données par exemple, on recherche pourtant ce type de capacité d'interprétation.



Comparaison des champs réceptifs CNN/Transformer. *Source :* AI Toolhouse Blog[28]

[28] TOSHNIWAL, Aditya. *ViT vs CNNs in AI Image Processing* [en ligne]. 15/05/24 [consulté le 20/12/25]. Disponible sur : https://blog.aitoolhouse.com/vits-vs-cnns-in-ai-image-processing/.

Au-delà de la simple détection, **il est donc parfois utile d'adopter un paradigme différent de la convolution pure : en l'occurrence, une solution a été apportée par les *transformers* en 2017**[29]. Initialement développés pour traiter le langage naturel, ils reposent sur **un principe de pondération dynamique inter-éléments**. Le langage présente en effet la même problématique que celle présentée ci-dessus : pour comprendre une phrase, il ne faut pas seulement regarder les mots directement voisins à un mot donné, mais bien tous les mots de la phrase, y compris les plus éloignés. En introduisant **un mécanisme dit « d'auto-attention », les *transformers* apportent une vraie rupture technologique**, en cela qu'ils sont capables de décider eux-mêmes quelles régions de l'entrée sont pertinentes pour une tâche donnée. En quelque sorte, ils transforment l'adjacence géographique en « adjacence de pertinence », et permettent de mettre en relation des régions de l'image éloignées, mais liées sémantiquement ou par d'autres liens latents. Si l'on dé-zoome encore en matière d'IA, il est possible de dire que les *transformers* ont réellement révolutionné le domaine – par exemple, ce sont eux qui ont ouvert la voie aux LLM (ChatGPT, Mistral, etc.) tels qu'on les connaît aujourd'hui.

En matière de vision, un certain nombre d'architectures basées sur les *transformers* ont donc été mises en place et comparées avec les CNN, pour différentes tâches. Il y a eu plusieurs approches d'intégration (par exemple en remplaçant directement les convolutions par des *transformers* dans un ResNet[30]), mais il faut mentionner en particulier les *« ViT »* (*« Vision Transformers »*[31]). Ils fonctionnent en transformant les images en petites sous-images (*« patches »*), qui sont ensuite traitées comme le seraient des éléments grammaticaux dans une phrase en langage naturel. En fonction des tâches, la performance des architectures *ViT* est très souvent supérieure à celle des CNN. Encore mieux : on peut fusionner les deux approches et créer des modèles hybrides[32].

Plus de *détails* seront donnés ci-après, mais Meta sort depuis 2021 **des versions de DINO**[33] (*« self-DIstillation with NO labels »*), qui est à la fois une méthode d'entraînement et un modèle de fondation basée sur les *ViT*. Cette technique permet de s'affranchir de la labellisation préalable de toutes les données d'entraînement ; elle obtient d'excellents résultats sur les tâches de vision classiques, sans pour autant avoir été entraînée spécifiquement à cet effet.

[29] « Attention Is All You Need ». *Wikipédia* [en ligne], 18/12/25 [consulté le 20/12/25]. Disponible sur : https://en.wikipedia.org/wiki/Attention_Is_All_You_Need.

[30] RAMACHANDRAN, Prajit & al. « Stand-Alone Self-Attention in Vision Models ». *NeurIPS* [en ligne], 2019. Disponible sur : https://proceedings.neurips.cc/paper/2019/hash/3416a75f4cea9109507cacd8e2f2aefc-Abstract.html.

[31] DOSOVITSKIY Alexey & al. « An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale ». *Arxiv* [en ligne], 03/06/21[consulté le 20/12/25]. Disponible sur : https://arxiv.org/abs/2010.11929.

[32] KORESH, Ella & al. « Unified CNNs and transformers underlying learning mechanism reveals multi-head attention modus vivendi ». *Physica A: Statistical Mechanics and its Applications* [en ligne], 15/05/25. Vol. 666. Disponible sur : https://www.sciencedirect.com/science/article/abs/pii/S0378437125001815.

[33] « DINOv3 ». *Meta* [en ligne]. Consulté le 20/12/25. Accès : https://ai.meta.com/dinov3/.

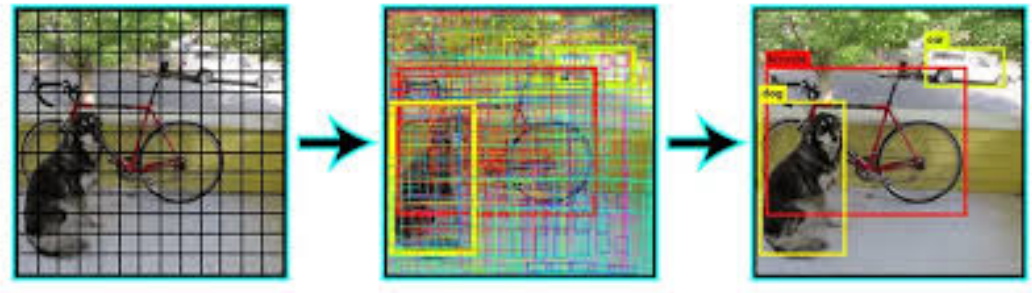
En bref : les réseaux de neurones convolutifs ont révolutionné la vision par ordinateur. Depuis 2012 et sur tous les cas d'usage classiques (segmentation, détection, classification, *tracking*, etc.), ils permettent d'obtenir de très bonnes performances grâce à l'intégration de techniques de convolutions – bien connues en traitement d'image standard – à l'intérieur de réseaux de neurones. **Cela permet d'apprendre automatiquement des filtres de convolution proches de l'optimalité** pour une tâche et un jeu de données fixés, mais présente le désavantage de ne considérer les images que comme des sous-ensembles locaux. Ce n'est pas forcément un problème et beaucoup de tâches de vision peuvent être très bien traitées ainsi. **Toutefois, une sophistication importante des modèles de vision est l'introduction du paradigme *transformer*, qui permet de changer totalement d'approche et d'appréhender des dépendances globales.** Sans perte de performances dans un cas général, cette technique permet de dépasser la simple observation automatique et ouvre la porte à une meilleure interprétation de l'image dans son ensemble – plus proche de ce que ferait le cerveau humain.

III. Mise en pratique : exemples d'applications

Les notions techniques présentées ci-avant peuvent donc maintenant être mises en application. Là encore, beaucoup de détails seront passés sous silence : l'objectif est de présenter des grands types de technologies et les éléments importants pour comprendre la mise en pratique et l'utilisation concrète de modèles de vision.

III.A. La gamme des modèles YOLO

Je commence donc avec le plus grand incontournable du sujet : la famille de modèles YOLO[34] (*« You Only Look Once »*). Introduite en 2015 et développée aujourd'hui en *open source* par l'entreprise Ultralytics[35], il s'agit d'une architecture de CNN conçue pour des tâches de détection d'objets sur images. Plusieurs améliorations ont ensuite été publiées (typiquement, une nouvelle version sort tous les ans), apportant des améliorations à une suite de modèles devenue absolument emblématique de la vision, surtout pour des applications embarquées. Et pour cause : au-delà d'une simple implémentation de CNN, **YOLO peut tourner en temps réel** (de quelques dizaines à une centaine de millisecondes par image) **grâce à un couplage de la localisation et de la classification des objets** – ce qui était fait, jusqu'ici, en deux étapes successives (voir R-CNN[36] et Fast R-CNN[37]). **YOLO permet donc de traiter entièrement la tâche de détection en une seule passe *« feed-forward »* du réseau :** c'est d'ailleurs de là qu'il tire son nom, car il ne « regarde qu'une fois » l'image.



Détection d'objets par un modèle YOLO. Source : Labellerr

[34] « You Only Look Once ». *Wikipédia* [en ligne], 09/11/25 [consulté le 20/12/25]. Disponible sur : https://en.wikipedia.org/wiki/You_Only_Look_Once.
[35] « Ultralytics » [en ligne]. Consulté le 20/12/25. Accès : https://www.ultralytics.com/ .
[36] GIRSHICK, Ross & al. « Region-Based Convolutional Networks for Accurate Object Detection and Segmentation" » *IEEE Transactions on Pattern Analysis and Machine Intelligence* [en ligne], 25/05/15, vol. 38 c[onsulté le 20/12/25]. Disponible sur : https://ieeexplore.ieee.org/document/7112511.
[37] GIRSHICK, Ross. « Fast R-CNN ». *Arxiv* [en ligne], 27/09/15 [consulté le 20/12/25]. Disponible sur : https://arxiv.org/abs/1504.08083.

Une idée centrale de YOLO est de diviser l'image en une grille de sous-images, et de les traiter les unes après les autres. Le modèle combine ensuite les prédictions obtenues (boîtes englobantes autour des objets détectés et probabilités de classification), pour constituer une prédiction globale sur toute l'image.

Au-delà de ça, l'architecture est relativement classique : couches de convolution, *pooling* et couches *fully-connected*, comme on a vu précédemment. On distingue tout de même 3 grands blocs dans l'architecture de la famille YOLO :

- **Le *« backbone »*.** C'est un premier ensemble de couches, qui sert à extraire les caractéristiques de l'image. En sortie du *backbone*, on obtient une représentation dite « latente » de l'image, c'est-à-dire un vecteur de nombres qui la représente de manière compacte et pertinente vis-à-vis de la tâche ;

- **Le « *neck* ».** C'est un deuxième ensemble de couches, un peu technique, dont la mission est de combiner les caractéristiques issues de différentes profondeurs du réseau. En effet, en fonction de la profondeur d'une couche donnée dans le *backbone*, les caractéristiques extraites ne sont pas les mêmes et peuvent témoigner de détails différents (objets de tailles variées, différentes perceptions des textures, etc.). En faire une combinaison permet quasi-toujours d'obtenir une meilleure performance : c'est le rôle du *neck* ;

- **Les « têtes de détection ».** Ce sont des couches très simples en sortie du *neck*, qui permettent d'obtenir les prédictions finales souhaitées : coordonnées des objets, probabilités de classes, scores de confiance, etc.

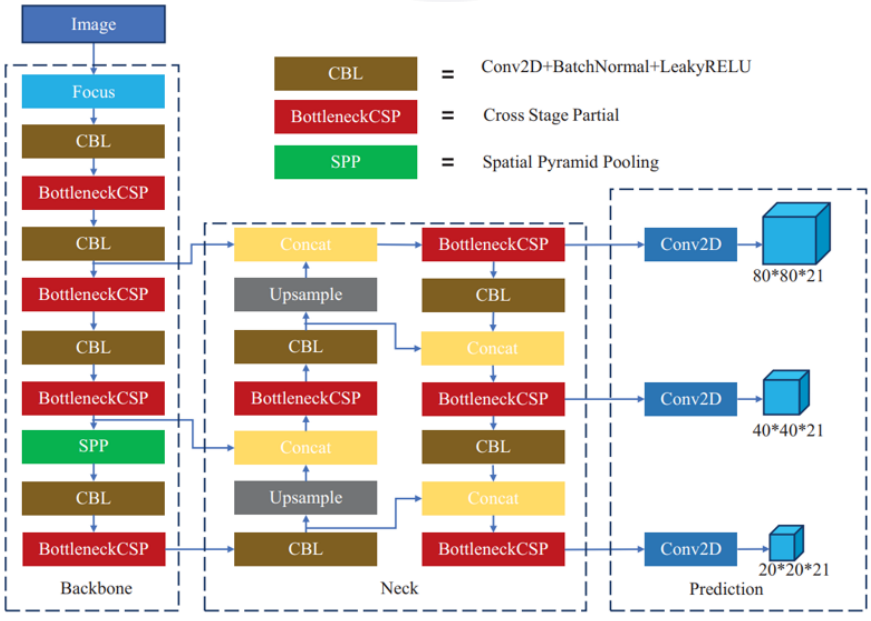


Schéma de l'architecture de YOLOv5*, les rectangles représentant des couches. Source :* ResearchGate

Historiquement, les versions de YOLOv1 à YOLOv8 étaient très similaires, et **leurs améliorations consistaient majoritairement à la sophistication de chacun des composants** précédents : plus grandes profondeurs, meilleure combinaison dans le *neck*, nouvelles fonctions de pertes pour mieux entraîner le réseau, et autres astuces techniques. Toutefois, les YOLO restaient toujours « bloqués » sur un paradigme convolutif, avec les désavantages présentés ci-avant. **C'est pourquoi les dernières versions de YOLO** (v11, v12, etc.) **intègrent désormais de plus en plus des mécanismes d'attention** inspirés des *transformers*. Cela permet de combiner le meilleur des deux approches : la convolution pour la simplicité et l'extrême pertinence locale, et la pondération dynamique inter-régions permise par les *transformers* pour améliorer la cohérence globale dans l'analyse de l'image.

Quoiqu'il en soit, les YOLO ont été et sont toujours industrialisés dans beaucoup d'applications, notamment embarquées, grâce à leur (de plus en plus relative) simplicité technique, leurs excellentes performances (voir la présentation des métriques ci-après)**, leur embarquabilité** (relativement peu de ressources nécessaires), **et surtout leur approche *« feed-forward »*** qui permet d'obtenir des inférences très rapides, quasiment en temps réel. De plus, les YOLO sont souvent relativement faciles à utiliser (Ultralytics met à disposition des API) et à *« fine-tuner »*, c'est-à-dire à spécifier à un contexte d'utilisation précis (sur des données spécifiques à un cas d'usage typiquement).

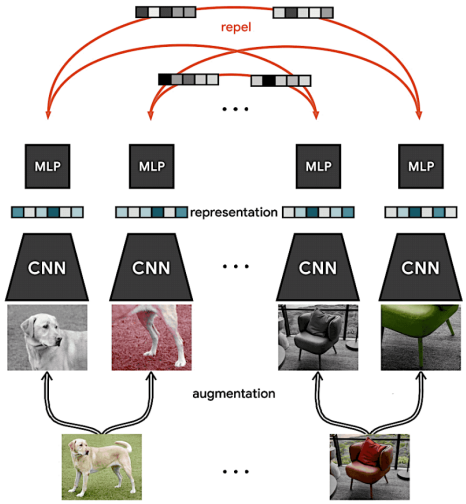
On retrouve ainsi cette famille de modèles dans la navigation autonome de véhicules, dans l'imagerie médicale, dans la sécurité (reconnaissance de personnes sur images caméra), dans l'industrie (détection de défauts sur pièces, comptage d'objets, vérification de conformités) et, bien évidemment, dans les applications militaires : ciblage, robotique autonome, détection d'intrusion et autres.

Détection automatique d'un tir missile grâce à un YOLO*. Source :* Medium

Si les dernières versions de YOLO incluent de plus en plus d'attention, ces modèles-là restent toutefois fondés sur la convolution. Or, comme expliqué précédemment, il est également possible de construire un modèle de vision directement sur le

paradigme *transformer* : typiquement les modèles de type *ViT*. L'approche DINO de Meta joue désormais un rôle très important dans l'écosystème de la vision, sans pour autant remplacer YOLO, qui reste encore extrêmement performant en vitesse d'inférence et en frugalité.

L'intérêt de DINO est ailleurs. **En réalité, ce n'est pas un modèle de détection au sens strict (comme YOLO), mais plutôt un modèle de fondation de représentations visuelles génériques** – c'est-à-dire une base de connaissances colossale de ce qu'est une image et de ce que sont les objets dans les images. Cette base-là a été obtenue par une technique d'apprentissage dite « auto-supervisée » (et également appelée DINO[38]), qui ne consiste pas à comparer les sorties du réseau avec la vérité attendue (comme on a fait jusqu'à maintenant), **mais plutôt à rapprocher, dans l'espace latent des couches neuronales, les représentations des entrées qui doivent y être similaires**. En pratique, on modifie artificiellement une même image pour en obtenir plusieurs, dont on sait donc par construction qu'elles représentent la même chose et dont il faut rapprocher les représentations. L'idée est de s'affranchir du besoin de connaître la vérité en sortie – la labellisation –, pour laisser le réseau construire lui-même des représentations pertinentes des objets. On le laisse comprendre qu'un chat ressemble un peu à un chien, qu'un chat est extrêmement différent d'un porte-avions, qu'un porte-avions ressemble un peu à une frégate, qu'une frégate est extrêmement différente d'un ballon de football, et ainsi de suite.



Principe d'apprentissage autosupervisé avec la technique SimCLR[39]. DINO *fonctionne différemment, mais l'idée fondamentale reste globalement la même. Source :* Towards Data Science

[38] « Vision transformer ». *Wikipédia* [en ligne], 18/12/25 [consulté le 20/12/25]. Disponible sur : https://en.wikipedia.org/wiki/Vision_transformer#DINO .

[39] CHEN, Ting ; KORNBLITH, Simon ; NOROUZI, Mohammad & HINTON, Geoffrey. « A Simple Framework for Contrastive Learning of Visual Representations ». *Arxiv* [en ligne], 01/07/20 [consulté le 20/12/25]. Disponible sur : https://arxiv.org/abs/2002.05709.

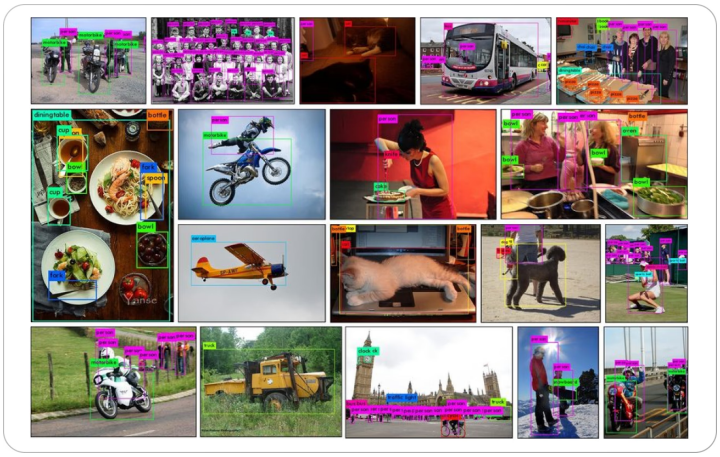
En implémentant cette technique d'apprentissage, Meta utilise la meilleure capacité d'intégration de connaissances cohérentes permise par les *transformers*, afin de **réduire la dépendance à des jeux de données massivement annotés et de généraliser la connaissance apprise dans des contextes opérationnels très variés**. En effet, une fois le modèle de fondation obtenu, il suffit de le *fine-tuner* dans le contexte et sur une tâche souhaitée, afin de profiter de ses connaissances extrêmement larges et de sa capacité de modélisation globale. On peut aussi utiliser les représentations de DINO en amont de modèles YOLO, voire directement en lieu et place du *backbone* (même si ce n'est pas du tout trivial) ; beaucoup de choses sont possibles… Sans présenter les caractéristiques d'une véritable rupture technologique, il est certain que **cette convergence architecturale vers les *transformers* et les modèles de fondation, y compris donc en matière de vision, illustre une tendance de fond assez générale en IA :** on cherche à construire des modèles « bibliothèques de connaissance », car, *in fine*, ils rendent plus de cohérence et de capacité de généralisation dans les tâches.

III.B. Le triptyque "données - pré-traitement - entraînement"

Jusqu'ici, nous avons beaucoup parlé d'architecture de réseaux. **Mais la performance d'un modèle n'est bien sûr pas que fonction de l'architecture :** elle dépend autant, voire davantage, des données qui sont utilisées pour son entraînement, et du paramétrage de cet entraînement. Qualité et quantité des données, pré-traitement, hyperparamètres : **ces trois éléments forment un triptyque primordial à considérer pour obtenir une performance satisfaisante**.

Tout d'abord concernant les données. Les explications se concentreront ici sur l'approche « standard » d'apprentissage supervisé (type CNN et YOLO), et laisseront de côté l'auto-supervisé (type DINO). Dans notre cas, on comprend intuitivement assez bien que certains critères soient importants pour une tâche de détection : il faut que les objets recherchés soient présents **en quantité et qualité** suffisante sur les images d'entraînement (sous différents angles, conditions d'éclairage, tailles, etc.), et il faut que **les labels soient suffisamment fiables** (l'opérateur qui a défini la vérité de référence a-t-il bien positionné les boîtes autour des objets ? A-t-il rangé les objets dans les bonnes catégories ?). Les modèles de vision sont typiquement pré-entraînés sur des *datasets* génériques et représentatifs d'une grande quantité d'objets et de situation ; **l'objectif étant ensuite que l'ingénieur construise son propre *dataset* spécifique sur lequel *fine-tuner*** (ajuster) le modèle pré-entraîné, sur sa tâche précise et dans son contexte visuel précis. S'il faut garder ces règles à l'esprit (équilibre, diversité, qualité, quantité des données) pour construire ces *datasets* spécifiques, elles sont globalement respectées dans les *datasets* de pré-

entraînement. Deux exemples très classiques sont ImageNet[40], qui a marqué un tournant en introduisant des millions d'images annotées sur des milliers de classes, et COCO[41], qui a intégré la détection multi-objets et la segmentation de scènes plus complexes. Ces deux jeux ne sont pas parfaits, mais ils fournissent déjà une grande base commune pour préparer et pré-entraîner des modèles transférables.



Exemples du dataset COCO. *Source :* COCO

De plus, avant d'être montrées au réseau pour l'entraînement, les images font quasi-systématiquement l'objet d'**un pré-traitement**. Cela peut servir à normaliser les couleurs, uniformiser la dimension des images, convertir des formats de couleurs, … stabiliser les données. Mais cela peut aussi servir à en améliorer la diversité et à améliorer la robustesse de l'entraînement. On pratique alors **la *« data augmentation »***[42], qui consiste à générer artificiellement de nouvelles images à partir des « vraies » images. Concrètement, on fait des rotations et des recadrages, des variations de luminosité, ou bien du masquage de régions de l'image. L'idée est d'enrichir le jeu de données d'entraînement grâce à un cumul de modifications élémentaires et aléatoires appliquées à chacune des images. Cela paraît désordonné et peu fiable, mais en pratique, **cela améliore très souvent la performance finale** du modèle sur la tâche souhaitée car cela permet de s'affranchir de certains biais de représentations. Ces techniques d'augmentation sont également **centrales dans les approches auto-supervisées** comme DINO : ce sont elles qui créent les images « semblables » dont il faut rapprocher les représentations latentes à l'entraînement.

[40] « ImageNet » [en ligne]. Consulté le 20/12/25. Disponible sur : https://www.image-net.org/.
[41] COCO. *COCO* [en ligne]. Consulté le 20/12/25. Disponible sur : https://cocodataset.org/.
[42] « Augmentation de données ». *Wikipédia* [en ligne], 09/09/25 [consulté le 20/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Augmentation_de_donn%C3%A9e.

Exemples de data augmentation*. Source :* Analytics Vidhya

Le dernier élément du triptyque repose sur des choix assez techniques mais vraiment cruciaux : **les paramètres de l'entraînement-même, dits « hyperparamètres »**. Combien d'images montre-t-on au réseau à chaque « boucle » d'entraînement ? Comment compare-t-on les prédictions du réseau avec les vérités attendues (quelle **fonction de perte**) ? Comment **optimise**-t-on les poids du réseau à chaque *backpropagation* ? Combien de temps doit durer l'entraînement (nombre de ***« batches »*** et ***« d'epochs »***) ? Les gradients doivent-ils avoir une faible ou une forte influence sur les poids du réseau à chaque *backpropagation* (quel ***« learning rate »***) ? Et ainsi de suite. Cela commence à devenir très technique, et il n'y a que relativement peu de règles générales. Ce qu'il faut retenir, c'est que ces paramètres doivent être absolument adaptés à chaque situation (cas d'usage, ensemble de données, tâche souhaitée, pré-entraînement ou *fine-tuning*), sous peine de ruiner totalement la performance du modèle.

En résumé, et c'était l'objectif de cette partie de le montrer : l'architecture d'un modèle de vision fixe son potentiel théorique et son type d'approche (convolution et/ou attention), mais c'est bien le triptyque données – pré-traitement – entraînement qui définit l'efficacité réelle du modèle prêt à l'emploi et sa capacité à généraliser.

III.C. Évaluer la performance : métriques et interprétations

Nous voilà avec un modèle dont l'architecture a été fixée, les images d'entraînement choisies et pré-traitées pour constituer un *dataset* fiable et robuste, et des hyperparamètres réglés correctement. Le modèle est donc désormais entraîné sur une tâche spécifique. **Il ne reste donc qu'à l'évaluer : comment peut-on savoir s'il fonctionne bien ? S'il est meilleur ou moins bon qu'un autre modèle ?** L'idée naïve (et plutôt bonne en première approche) serait de compter les nombres de prédictions correctes et incorrectes, et de les comparer avec ceux d'autres modèles. Toutefois, les exemples ci-dessous montreront que beaucoup de nuances doivent être prises en compte : **il y a donc besoin d'utiliser des métriques différentes,**

qui renseignent chacune sur un aspect de la performance. Seul l'ensemble de ces métriques[43] permet vraiment d'évaluer un modèle.

Avant de présenter les métriques, il faut introduire une terminologie commune cruciale : les catégories « vrais positifs » / « vrais négatifs » / « faux positifs » / « faux négatifs ». Concrètement, pour des tâches de classification ou de détection par exemple, il faut imaginer un objet dont la vérité est une classe donnée (« drone » par exemple). Le modèle va donner une prédiction (« drone » si la prédiction est correcte, mais autre chose sinon : « chaton » ou « banane » par exemple). On distingue alors 4 possibilités :

- **Vrai positif (« TP ») :** l'objet est correctement détecté et correctement classifié (un drone est détecté dans l'image, et c'est effectivement la vérité) ;
- **Faux positif (« FP ») :** le modèle détecte un objet et le classifie, mais il y a une ou plusieurs erreurs (un chaton est détecté dans l'image, mais en réalité il n'y a rien ; un chaton est détecté dans l'image, mais il aurait fallu détecter un avion ; un chaton est détecté à gauche de l'image, alors qu'il se situe en réalité à droite) ;
- **Faux négatif (« FN ») :** un objet d'une certaine classe est réellement présent dans l'image, mais le modèle ne le détecte pas ;
- **Vrai négatif (« TN ») :** aucun objet n'est présent en réalité, et le modèle ne détecte effectivement rien

Ces 4 quantités sont la base des métriques classiques : avant toute chose, il faudra donc souvent compter chacun des TP/FP/FN/TN, sur le jeu de validation, à partir desquels on pourra construire les métriques. Ces métriques classiques sont représentées dans ce que l'on appelle la table de confusion, ci-dessous :

		Vraie condition	
		Condition positive	Condition negative
Détection	Détecté positif	TP	FP
	Détecté negatif	FN	TN

Tableau récapitulatif. Source : ResearchGate

[43] MALVIYA, Nikita. « Different Evaluation metrics for Computer Vision tasks ». *Medium* [en ligne], 03/08/23 [consulté le 20/12/25]. Disponible sur : https://medium.com/@nikitamalviya/different-evaluation-metrics-for-computer-vision-tasks-83feb9a1041b.

- **La précision.** C'est la première métrique, et la plus basique. Elle mesure très simplement la proportion d'objets correctement détectés parmi tous les objets détectés : TP / (TP + FP). **Plus elle est élevée, et moins le modèle fait de fausses alertes.** Il faut notamment la maximiser lorsqu'une détection erronée est très grave (identifier une mauvaise cible par exemple), c'est-à-dire dans les cas où il faut absolument éviter les faux positifs ;

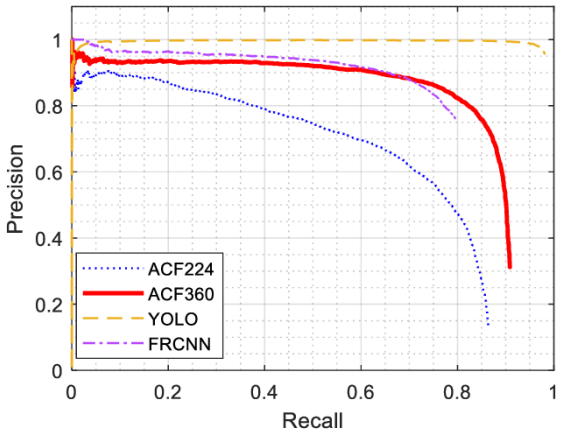
- **Le rappel (*« recall »*).** Cependant, dans certains cas d'usage où les vrais positifs sont rares, la précision est une mauvaise métrique. Par exemple pour la détection d'anomalies sur des images médicales : si 99 % des patients sur Terre sont sains, le modèle pourrait prédire dans 100 % des cas qu'il n'y a pas d'anomalie… et ne se tromper que dans 1 % des cas. Sauf qu'il ne servirait alors à rien, puisqu'il manquerait systématiquement tous les cas anormaux ! On introduit donc la notion de rappel, calculée comme TP / (TP + FN). **Concrètement, maximiser le rappel signifie accepter davantage de fausses alertes (faux positifs) pour être sûrs de ne rater aucun vrai positif ;**

- **Le score F1.** En pratique, précision et rappel sont donc souvent antagonistes, puisqu'il est complexe d'augmenter l'un sans diminuer l'autre. **On recherche donc traditionnellement un compromis :** « où placer le curseur pour ne pas rater de vrais positifs, sans déclencher trop de fausses alertes ? ». Une solution populaire pour quantifier ce compromis est **le score F1, calculé comme la moyenne harmonique des 2 métriques :** 2 * Précision * Rappel / (Précision + Rappel). Lorsque les classes d'objets sont déséquilibrées par exemple, il est souvent intéressant de rechercher un score F1 assez élevé ;

- **L'IoU (*« Intersection over Union »*).** En parallèle de l'évaluation de la classification-même, il faut également savoir **mesurer la qualité de la localisation spatiale des objets détectés**. Une façon classique est de regarder l'IoU, calculée à partir des boîtes englobantes prédites et attendues (vérité) autour des objets à détecter. L'intersection représente l'aire du chevauchement des deux boîtes, tandis que l'union représente l'aire totale recouverte par les deux. Calculer leur ratio permet de quantifier la superposition des deux boîtes : 0 % signifiant donc qu'elles ne sont pas du tout au même endroit (donc prédiction très mauvaise), et 100 % signifiant qu'elles se recouvrent totalement (donc prédiction parfaite). Une détection est souvent considérée comme correcte si son IoU dépasse 50 %.

Illustration de la notion IoU. *Source :* PyImageSearch

- **La *mAP (« mean Average Precision »)*.** Pour une classe donnée et un seuil d'IoU fixé, on peut donc tracer une courbe précision-rappel[44] en faisant varier le seuil de confiance des prédictions. S'il est élevé, on tendra vers une grande précision et un faible rappel (donc peu de détections, mais d'excellentes détections), tandis que s'il est faible, on tendra vers une faible précision et un grand rappel (beaucoup de prédictions, mais beaucoup de fausses alertes). **La courbe précision-rappel permet de visualiser cet arbitrage.** On peut, de plus, calculer l'aire sous sa courbe : c'est ce que l'on appelle *« l'Average Precision »*, qui peut être moyennée sur toutes les classes (toutes les courbes) pour donner la *« mean Average Precision »*. Ces deux grandeurs quantifient l'inspection visuelle que l'on peut faire de la courbe précision-rappel, qui peut devenir difficile lorsque l'on cherche à optimiser finement les métriques. **La mAP@[0.50;0.95]** (c'est la mAP moyennée sur plusieurs seuils d'IoU entre 50 % et 95 %) **est aujourd'hui la métrique de référence pour comparer les modèles de détection d'objets.**



Courbe précision-rappel pour différents modèles, dont un YOLO. *Source :* ResearchGate

Il y a d'autres métriques assez importantes : la courbe ROC, la MSE et la RMSE, le PSNR, etc. Mais elles restent relativement marginales comparées à celles, vraiment fondamentales, présentées ci-avant.

[44] « Précision et rappel ». *Wikipédia* [en ligne], 10/12/25 [consulté le 20/12/25]. Disponible sur : https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel.

Ce qu'il faut garder en tête, c'est que l'évaluation d'un modèle est bien plus technique qu'elle ne pourrait paraître : en fonction des cas d'usage, de la tâche à optimiser ou du profil de risque (accepte-t-on les fausses alertes ? Les erreurs ?), le réglage du modèle ne se fait pas de la même manière et peut avoir de lourdes conséquences : en matière militaire par exemple (et pour schématiser à l'extrême), on ne peut pas se permettre d'engager des faux positifs, et il ne faut pas laisser passer les faux négatifs.

III.D. Exemples d'applications

La théorie est terminée ! C'était peut-être un peu lourd, mais nous avons ainsi fait un tour d'horizon des principales notions techniques à garder en tête pour comprendre comment fonctionne un modèle de vision par ordinateur moderne :

- Performances comparées entre *deep learning* et traitement d'image classique ;
- Motivations pour les applications de défense ;
- Réseaux de neurones et apprentissage ;
- Images numériques et traitements statistiques ;
- Convolution et techniques associées ;
- Réseaux convolutifs ;
- « Paradigme *transformer* » et application à la vision ;
- La famille YOLO ;
- DINO ;
- Le triptyque « données – pré-traitement – entraînement » ;
- Les métriques d'évaluation.

Désormais, nous allons donner quelques illustrations concrètes, **en restant sur de l'utilisation basique de modèles, avec les API standards et des cas d'usage très simples. L'objectif est simplement de donner des exemples** d'application de modèles d'IA pour la vision, de regarder leurs métriques de performances sur des cas simples, et de mettre un peu en pratique ce qui a été présenté ci-avant : **on n'y parlera pas de programmation Python, donc aucun pré-requis n'est nécessaire**. Nous proposons d'illustrer trois cas d'usage militaires typiques (qui se recoupent un peu) : de la détection d'objets sur images (exemples : lutte anti-drones), de la navigation autonome (exemple : segmentation de paysage pour véhicules terrestres), et de l'analyse d'imagerie satellite (exemple : suivi de navires).

III.D.1. Détection de cibles

Pour cette première illustration, **nous utilisons un YOLOv8** (par l'intermédiaire de l'API d'Ultralytics)**, ainsi qu'un *dataset* annoté de drones, trouvé sur Kaggle**[45]. Le modèle YOLOv8 est pré-entraîné sur des *datasets* génériques et très complets, mais pas du tout spécifiques à ce cas d'usage : en effet, l'objectif est d'apprendre à détecter des drones sur des images. Nous commençons donc par *fine-tuner* ce modèle sur un sous-ensemble du jeu de données. L'entraînement consiste à montrer ces images-là au modèle et à comparer sa prédiction des emplacements des drones avec leurs emplacements réels - connus donc, puisque les données sont annotées. À partir de la différence entre les prédictions et la vérité, on peut faire la *backpropagation* et ajuster automatiquement les poids pré-réglés de manière générique. Ici, le choix est fait de ne pas augmenter les données, car le *dataset* est relativement simple et le modèle devrait très bien fonctionner du premier coup. Concernant les hyperparamètres, nous utilisons un réglage standard : sans entrer dans les détails, dans la mesure où les données sont propres et assez peu hors du commun (environnement, luminosité, détails assez normaux), le modèle proposé par Ultralytics devrait suffire.

```
Validating /content/runs/detect/yolo_no_aug7/weights/best.pt...
Ultralytics 8.3.240 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 44/44 6.7it/s 6.6s
                   all        347        369      0.893      0.791      0.884      0.511
Speed: 0.4ms preprocess, 3.4ms inference, 0.0ms loss, 2.4ms postprocess per image
Results saved to /content/runs/detect/yolo_no_aug7
Ultralytics 8.3.240 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 1061.4±724.3 MB/s, size: 99.9 KB)
val: Scanning /root/.cache/kagglehub/datasets/muki2003/yolo-drone-detection-dataset/versions/1/drone_dataset/valid/labels.cache... 347 images, 0 backgrounds, 0 corrupt: 100% ━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 22/22 3.1it/s 7.1s
                   all        347        369      0.889      0.789      0.884      0.512
Speed: 2.1ms preprocess, 5.1ms inference, 0.0ms loss, 2.0ms postprocess per image
Results saved to /content/runs/detect/val
```

Le fine-tuning *s'est bien passé*

Puisque notre ensemble d'entraînement n'était qu'un sous-ensemble du *dataset*, il reste fatalement un autre sous-ensemble annoté jamais vu par le modèle : il va servir pour l'évaluer. Cela consiste à figer les poids (on arrête la *backpropagation*), afin de seulement prédire les localisations sur de nouvelles images, et de regarder, en métriques, si elles sont justes. Nous obtenons le tableau de métriques ci-dessous :

```
Ultralytics 8.3.240 🚀 Python-3.12.12 torch-2.9.0+cu126 CUDA:0 (Tesla T4, 15095MiB)
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 896.0±655.4 MB/s, size: 152.1 KB)
val: Scanning /root/.cache/kagglehub/datasets/muki2003/yolo-drone-detection-dataset/versions/1/drone_dataset/valid/labels.cache... 347 images, 0 backgrounds, 0 corrupt: 100% ━━━━
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ━━━━━━━━━━━━ 22/22 3.2it/s 6.8s
                   all        347        369      0.889      0.789      0.884      0.512
Speed: 1.6ms preprocess, 3.4ms inference, 0.0ms loss, 2.1ms postprocess per image
Results saved to /content/runs/detect/val4
Résultats fine-tuning sans augmentation :
mAP@0.5        : 0.884
mAP@0.5:0.95   : 0.512
Précision      : [    0.88926]
Rappel         : [    0.78862]
```

Nos métriques sur le sous-ensemble d'évaluation "drones"

[45] MUKILAN, S. « YOLO Drone Detection Dataset ». *Kaggle* [en ligne]. 2023 [consulté le 21/12/25]. Disponible sur : https://www.kaggle.com/datasets/muki2003/yolo-drone-detection-dataset/data.

N'entrons pas dans une comparaison exhaustive avec ce que pourraient être des plages de valeurs acceptables pour un modèle industrialisable, mais en général, on considère qu'une mAP@0.5 supérieure à 0,85 est bonne. Idem pour la précision et le rappel : ces valeurs semblent acceptables.

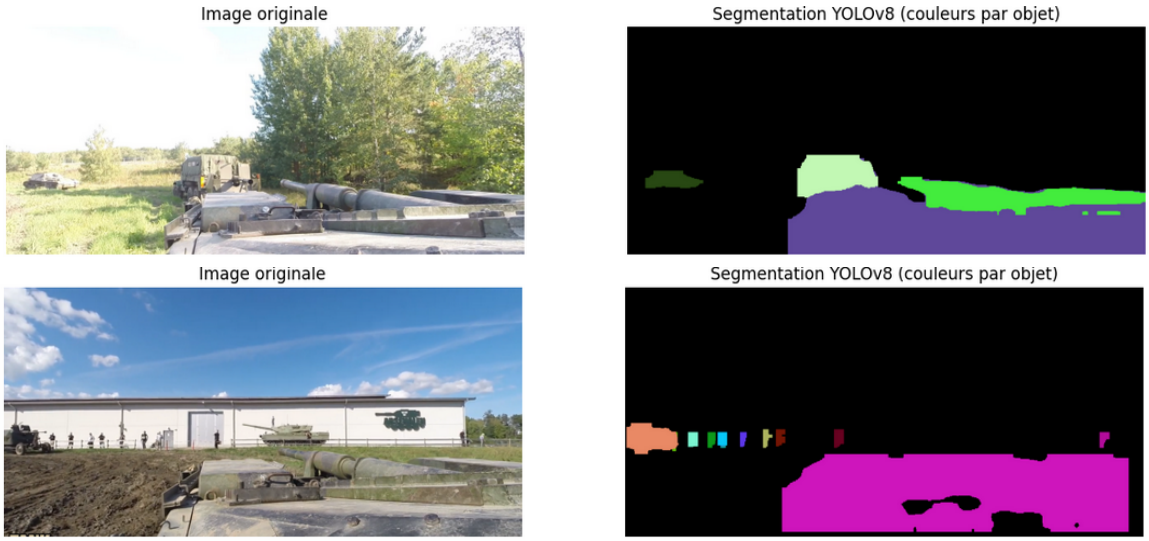
Visuellement, on peut se convaincre que le modèle fonctionne : ci-dessous, 3 images de test sont affichées avec 2 rectangles. Le rectangle vert, c'est la vérité terrain, c'est-à-dire l'annotation manuelle d'un opérateur humain. Le bleu, c'est la prédiction du modèle : l'objectif est qu'elle se superpose au carré clair. **La superposition des rectangles est importante… mais les images sont extrêmement simples :** ciel bleu uniforme, drones reconnaissables, aucun camouflage, conditions météo parfaites. **Naturellement, un cas militaire général est bien plus complexe et la chaîne doit être perfectionnée** (données d'entraînement, hyperparamétrage, *data augmentation*, etc.).

Illustration de la performance de notre modèle fine-tuné *pour détecter des drones dans un cas très simple*

III.D.2. Segmentation

Pour ce deuxième exemple, nous utilisons également un YOLOv8. Mais cette fois, nous prenons une version conçue spécifiquement pour faire de la segmentation, **c'est-à-dire découper l'image en régions censées avoir une même cohérence géographique ou sémantique**. Concrètement : découper le ciel du sol, découper un véhicule de la route sur laquelle il se trouve, découper le fusil de l'opérateur militaire qui le porte, découper les arbres dans une forêt, etc. Ce type de tâche est souvent utile pour concevoir des algorithmes de navigation autonome, car il permet de classifier pixel par pixel l'environnement observé par le véhicule, estimant donc l'espace navigable. C'est tout à fait complémentaire avec la détection d'objets, illustrée ci-avant.

Pour se mettre dans un contexte de drone terrestre autonome, nous prenons quelques images extraites de vidéos YouTube[46] [47] et représentant des chars roulant sur des chemins. Sans *fine-tuning* particulier cette fois, nous appliquons le modèle très simplement pour obtenir les résultats ci-dessous.



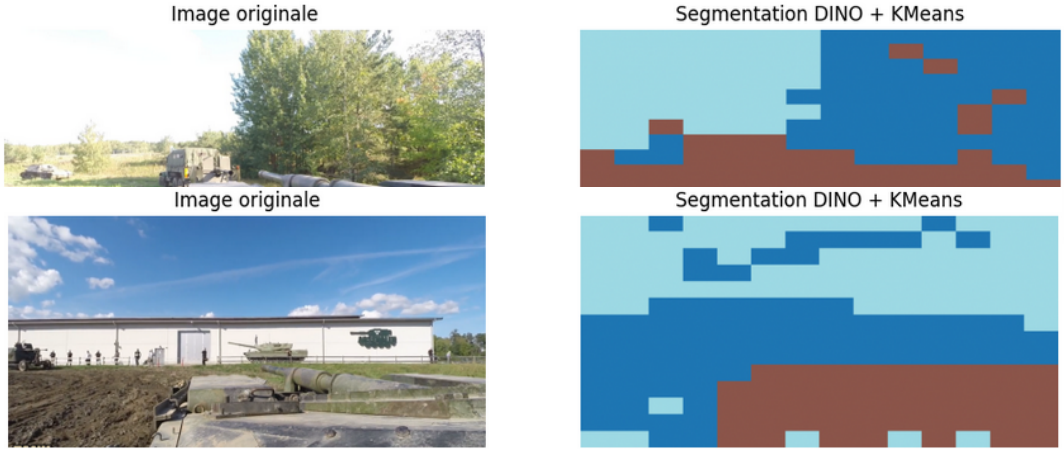
Résultats de la segmentation de nos images avec YOLOv8

Pour évaluer quantitativement la performance de la segmentation, il faudrait utiliser une métrique du type IoU. Cependant, nous n'avons pas de vérité absolue concernant les zones de segmentation, donc une étape préliminaire serait de les définir à la main : ce serait long et fastidieux, et ce n'est pas l'objectif de cette illustration. On comprend toutefois l'idée de la segmentation : le modèle, même dans sa version générique, arrive globalement à comprendre où se trouve le premier plan (en bas à droite), où se trouvent les autres véhicules (sur la 1ère image), et où se trouvent de potentiels obstacles (des opérateurs humains sur la 2e image). **Cette segmentation n'est quand même pas excellente ; on ne pourrait probablement pas l'exploiter en l'état.** Mais dans la mesure où nous n'avons pas spécifiquement entraîné le modèle pour ce contexte, **cet exemple montre un premier résultat brut déjà intéressant**.

Nous pouvons aussi essayer une autre technique : à partir des *features* extraites par DINOv3, trier les groupes de pixels en classes (avec des techniques de *clustering* classiques - ce n'est pas le sujet de cet article). **La résolution de cette segmentation est mauvaise car limitée par la taille des patchs pris par DINO**

[46] Tankspotting. *BMP-1 IFV Forest Ride - Tank Commander Seat PoV HeadCam* [vidéo en ligne]. 25/05/24 [consultée le 21/12/25]. Disponible sur : https://www.youtube.com/watch?v=KNMHxc8WSTY.
[47] Tankspotting. *Stridsvagn 103 (S-Tank) POV Ride Full Video - Action Set And Preparationing - Arsenalen* [vidéo en ligne]. 29/11/19 [consultée le 21/12/25]. Disponible sur : https://www.youtube.com/watch?v=ogTcL6ukZ7c.

(car, pour rappel, c'est une approche *transformer*)**, ce qui rend la comparaison difficile avec la segmentation YOLO. Toutefois, on remarque déjà des différences :** ici, le modèle a-t-il vraiment correctement identifié ce qui était au premier plan ?



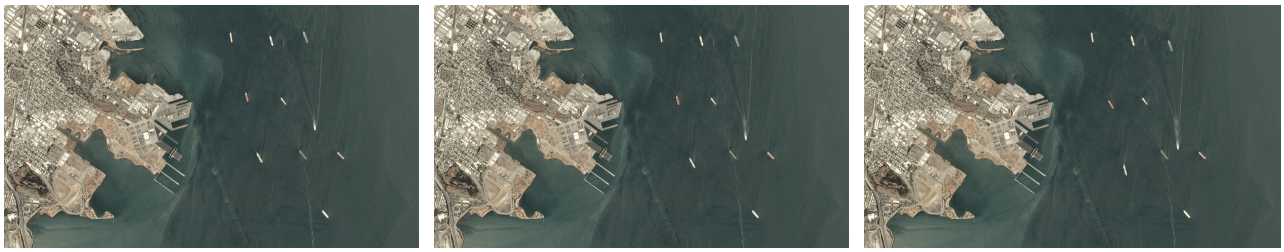
Résultats de la segmentation de nos images avec DINOv3 + KMeans

III.D.3. Suivi d'imagerie satellite

Ce qui est intéressant, maintenant, c'est de voir que l'on peut construire quelque chose au-dessus de la simple détection. L'imagerie satellite est un bon exemple pour cela. En effet, dans la mesure où la détection des objets peut être automatisée, on peut mettre en place des algorithmes de suivi automatiques. Par exemple, il devient possible d'automatiser la veille d'une même région, à chaque passe du satellite au-dessus de cette dernière : y a-t-il eu des changements dans les objets détectés dans la région ? Comment peut-on analyser les mouvements ? Présentent-ils des dangers qu'il faut remonter à un opérateur humain ? Et ainsi de suite.

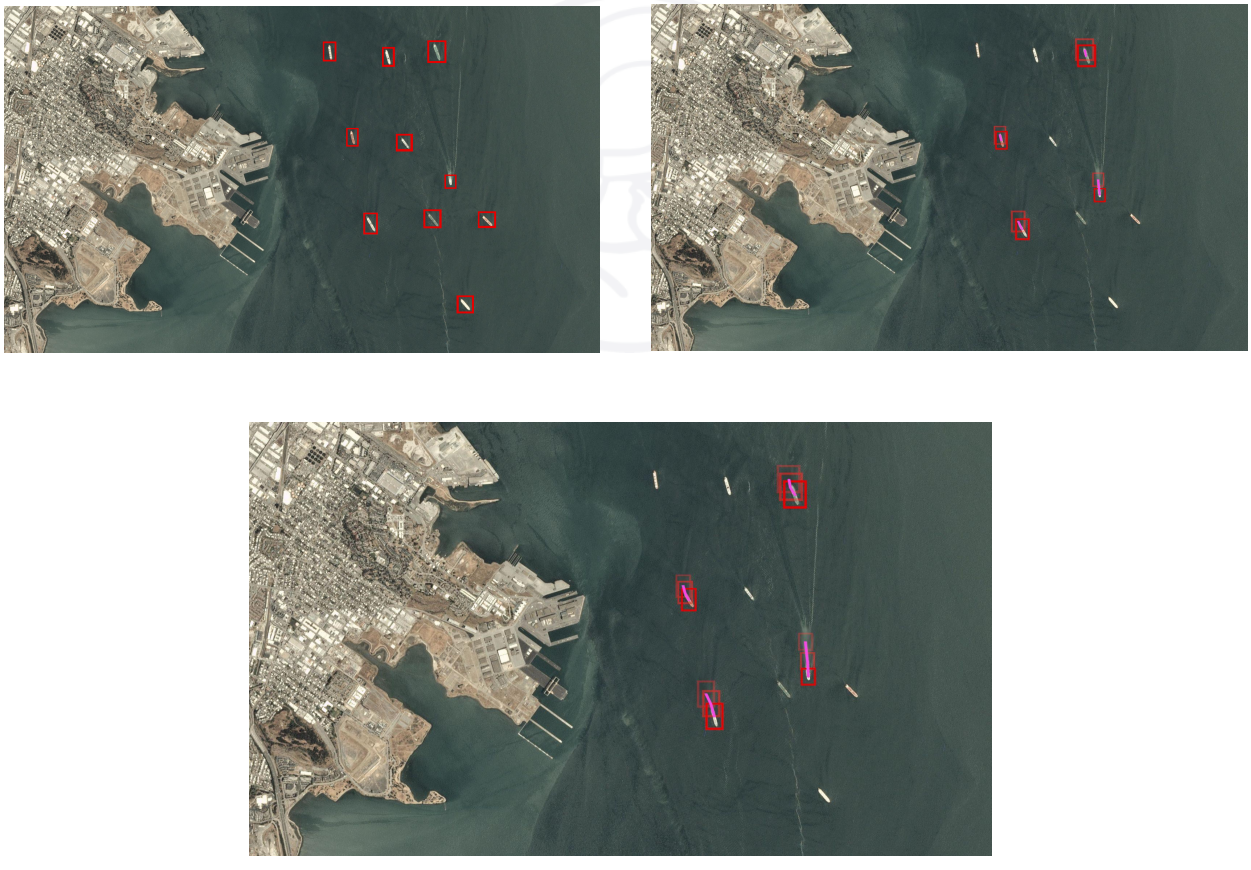
Pour illustrer, nous nous basons sur une suite d'images satellites[48] prises au-dessus d'un même port, mais à 3 moments différents :

[48] RHAMMELL. « Ships in Satellite Imagery ». *Kaggle* [en ligne]. 2018 [consulté le 21/12/25]. Disponible sur : https://www.kaggle.com/datasets/rhammell/ships-in-satellite-imagery/data.



Le même endroit, mais à 3 moments différents (dans l'ordre chronologique de gauche à droite)

Lors de la première passe de satellite, nous demandons simplement au modèle de détecter tous les bateaux. À la passe suivante, nous pourrons ainsi lui demander de ne mettre en évidence que les bateaux qui auront bougé par rapport à la première passe. Et ainsi de suite. Avec des calculs très simples, on peut aussi mettre en évidence la trajectoire des bateaux, et donc éventuellement prédire les trajectoires suivantes (avec d'autres systèmes d'IA, cela s'appelle le *"tracking"*). **Il est donc ainsi possible d'automatiser la veille stratégique ou opérative sur une zone, d'analyser les mouvements de matériels ou de troupes, et de ne faire remonter à l'opérateur humain que l'information urgente et/ou pertinente qui le justifie.**



Sur la première photo, tous les bateaux sont détectés. Sur les 2e et 3e photos, l'algorithme ne montre plus que les bateaux qui ont bougé entre temps. Une ligne rose retrace leurs parcours et peut être éventuellement utilisée par la suite pour prédire leurs trajectoires futures

IV. Et maintenant ?

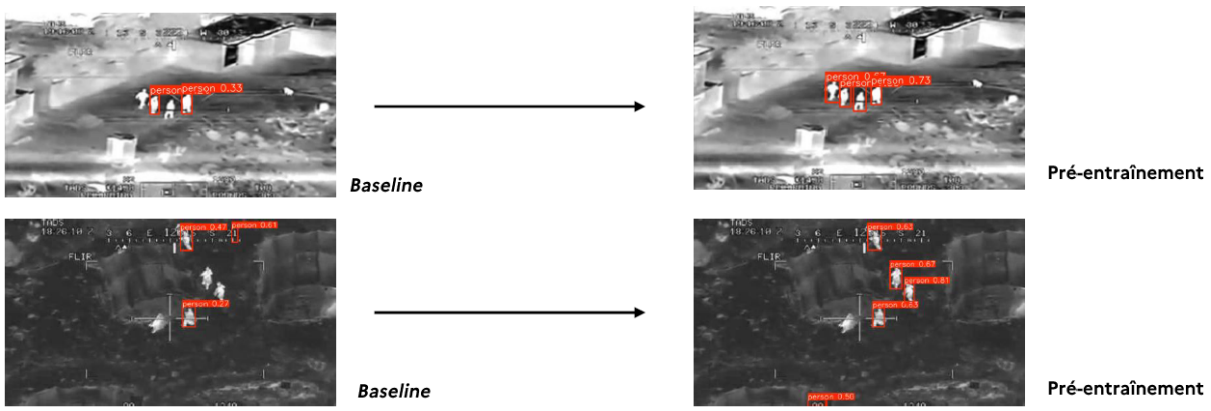
Il semble enfin que ce tour d'horizon des modèles de vision doive être complété par un panorama des défis actuels. Dans le passé, plusieurs améliorations technologiques ont permis de résoudre un certain nombre de problématiques, mais certaines autres demeurent : voici un mot rapide sur celles qui semblent aujourd'hui les principales pour les applications de défense.

IV.A. Quelques défis techniques actuels

En premier lieu, les applications défense génèrent plusieurs contraintes « dures ». Par exemple, la diversité des environnements, les conditions potentiellement extrêmes (température, pression, etc.) peuvent avoir un effet sur les capteurs, mais aussi sur la qualité des images à traiter : luminosités particulières, effets météo (brouillard par exemple.), contrastes inhabituels, nuit, camouflages et leurres, etc. De la même manière, certains cas d'usage nécessitent l'utilisation de longueurs d'ondes différentes du visible. L'expérience montre que les modèles classiques doivent être adaptés, et qu'un *fine-tuning* simple peut ne pas suffire pour maintenir la performance. Adapter les algorithmes pour qu'ils restent performants dans ces conditions est clef.

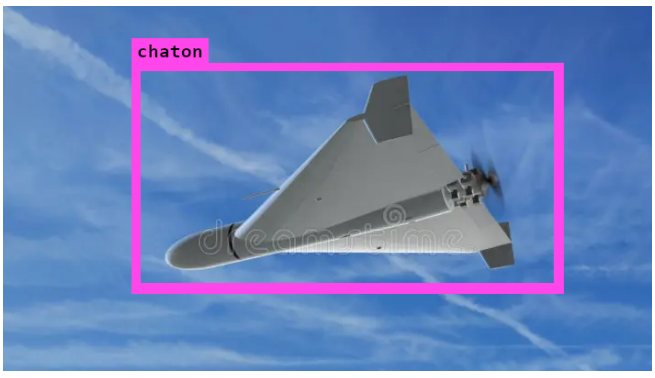
En parallèle, on pense également à **améliorer encore la performance des modèles**. Au-delà des images simples, une piste est d'**approcher la vision sous un prisme multimodal** : images, vidéos, capteurs thermiques, capteurs lidar, données sonores, … on peut essayer de fusionner différents types de données afin de raffiner davantage la tâche de vision « pure » que l'on souhaite. Finalement, n'est-ce pas ce que notre cerveau fait aussi ? Par exemple, pour localiser un chat dans notre champ de vision, on peut ouvrir les yeux, mais aussi se rappeler d'où il était il y a 2 secondes, ou encore écouter d'où provient le miaulement…

De plus, un certain nombre d'applications défense nécessitent d'embarquer directement les modèles – typiquement pour faire de la navigation autonome. Il y a donc une contrainte de puissance disponible et de bande passante : on comprend bien qu'un drone *low cost* n'a pas la même puissance de calcul qu'un ordinateur dans un centre de commandement. Il faut donc optimiser les architectures, réduire leur taille, privilégier des traitements simples, … bref, rechercher ce que l'on appelle **la frugalité**. C'est encore un sujet aujourd'hui.



Exemple avant/après amélioration d'un modèle de vision infrarouge. Source : Projet personnel

Un second élément concernant la performance est **le jeu de données d'entraînement**. On a vu que leur qualité, quantité, fiabilité, représentativité, etc. étaient primordiales pour bien entraîner un modèle, mais nous pourrions ajouter une autre dimension : **la sécurité des données**. En effet, si l'on ne maîtrise pas son jeu de données, des acteurs malveillants peuvent l'utiliser comme une porte dérobée pour induire le modèle en erreur ; **cela s'appelle du *« data poisoning »***[49] (et plus largement de *« l'adversarial machine learning »*). Imaginez que quelqu'un a remplacé à votre insu tous les labels « drones » par des labels « cailloux ». Le modèle deviendra absolument inutile : pas parce qu'il aura été mal entraîné (techniquement, il pourra être un excellent détecteur de « cailloux »), mais simplement parce que ses labels seront absurdes. À l'heure où les données de qualité militaire sont encore rares, et où il est courant d'entraîner des modèles (y compris professionnels) sur des jeux de données téléchargés en ligne, **il est donc important de se poser la question de leur maîtrise, de leur provenance et de leur fiabilité, avant de les utiliser**.



Exemple de data poisoning *: et si tous les drones Shahed étaient labellisés "chaton" ?*

[49] KRANTZ, Tom & JONKER, Alexandra. « What is data poisoning? ». *IBM* [en ligne]. Consulté le 21/12/25. Disponible sur : https://www.ibm.com/think/topics/data-poisoning.

Enfin, nous avons évoqué les *ViT* et DINO, mais on peut voir encore plus loin. L'approche *transformer* permet certes de traiter des images, **mais si l'on veut une approche encore plus « englobante », on peut penser à traiter des vidéos**. Un exemple (qui n'est probablement pas le plus pertinent, et certainement déjà dépassé par l'état-de-l'art) est VGGT[50], qui permet directement de reconstituer un environnement 3D à partir d'une vidéo. Pour les connaisseurs, c'est la même fonction que la photogrammétrie classique type COLMAP, mais en beaucoup plus rapide et efficace (notamment car c'est en *feed-forward* et non en passes d'optimisation). Peut-être qu'explorer ce type de tâches de compréhension 3D pourrait ouvrir des pistes en matière de « simple vision » ?

Ces éléments ne sont probablement pas exhaustifs, mais semblent tous importants à considérer pour comprendre où l'on en est, aujourd'hui, en matière de conception de modèles de vision. Comment adapter les modèles à des applications embarquées, dans des conditions visuelles extrêmes et très spécifiques aux applications militaires, en recherchant la performance état-de-l'art, la sécurité, la robustesse et la fiabilité ? Ces sujets sont encore ouverts.

IV.B. Enjeux doctrinaux et position française

L'objectif de cet article n'est pas de mener une réflexion transversale ni d'intégrer les notions techniques dans une étude de concepts opérationnels. Toutefois, il est difficile de parler des défis posés par cette technologie sans écrire un mot sur les risques éthiques et la doctrine.

De manière générale, l'IA pour la défense pose de vraies questions éthiques : qui est responsable ? Peut-on vraiment comprendre l'IA ? Comment exploiter son potentiel opérationnel tout en respectant le droit international et les fondements éthiques de la pensée française ? S'autorise-t-on à employer des systèmes d'IA qui pourraient prendre seuls des décisions ?

Pour réfléchir à ces questions et élaborer une doctrine nationale, **un comité d'éthique de la défense a été créé en 2019**. Il a notamment rendu 2 avis concernant le sujet de l'IA : l'un en 2021 sur l'intégration de l'autonomie dans les systèmes d'armes létaux[51], et l'autre en 2025 sur l'usage des technologies d'IA par

[50] WANG Jianyuan & al. « VGGT: Visual Geometry Grounded Transformer ». *Arxiv* [en ligne] 14/03/25. Disponible sur : https://arxiv.org/abs/2503.11651

[51] Comité d'éthique de la défense. *Avis sur l'intégration de l'autonomie dans les systèmes d'armes létaux*. 29/04/21. 47 pages. Disponible sur : http://www.defense.gouv.fr/sites/default/files/ministere-armees/20210429_Comit%C3%A9%20d%27%C3%A9thique%20de%20la%20d%C3%A9fense%20-%20Avis%20int%C3%A9gration%20autonomie%20syst%C3%A8mes%20armes%20l%C3%A9taux.pdf.

les forces armées[52]. Un élément central à retenir de ces avis, c'est la distinction de concepts entre **les SALA** (« Systèmes d'Armes Létaux Autonomes ») **et les SALIA** (« Systèmes d'Armes Létaux Intégrant de l'Autonomie »). **Les premiers, les SALA, sont par définition capables d'employer la force létale sans supervision humaine : pour cette raison – c'est contraire aux principes éthiques fondamentaux en France –, la doctrine officielle est de se les interdire.** En revanche, les SALIA ne font « qu'intégrer » de l'IA dans un cadre d'emploi clairement défini et validé par un opérateur, qui en porte lui-même la responsabilité. Dès lors, l'IA devient un simple outil, et même s'il prend une décision d'engagement, c'est toujours dans le cadre d'une chaîne de commandement, d'une supervision, et d'une responsabilité humaines. **Les SALIA ne sont donc pas interdits en France ; on considère au contraire que c'est le bon intermédiaire** pour profiter des avantages opérationnels et « rester dans la course », sans enfreindre de lignes rouges morales.

Tous les pays n'ont pas de doctrine si claire et si tranchée, et l'on a déjà observé sur certains théâtres des utilisations de l'IA que l'on pourrait qualifier de controversées. **En France toutefois, le principe est clair : malgré le fait qu'il puisse être défaillant dans la chaîne de contrôle, « l'humain doit rester dans la boucle »**, et l'IA ne doit ni définir, ni modifier une mission de manière autonome. C'est un outil au service de l'humain.

Pour revenir au sujet, cette réflexion doctrinale semble en fait essentielle pour employer les technologies de vision de la bonne manière. Une fois qu'une cible est détectée et classifiée par l'IA, l'engagement peut ensuite ne relever que de systèmes tout à fait déterministes (donc pas de l'IA), si bien que le vrai enjeu semble d'arriver à ne pas cibler n'importe quoi ou n'importe qui. Or, puisqu'il est par exemple nécessaire de choisir un compromis précision-rappel, on comprend bien que les erreurs de détection et de classification sont inévitables. **Même avec un excellent modèle, la question de la doctrine est donc incontournable :** si jamais une erreur de vision est faite par un modèle d'IA, il est moralement impossible de lui en faire porter la responsabilité. **L'humain doit donc nécessairement rester au centre.**

[52] Comité d'éthique de la défense. *Avis sur l'usage des technologies d'intelligence artificielle par les forces armées*. 14/01/25. 25 pages. Disponible sur : http://www.defense.gouv.fr/sites/default/files/ministere-armees/20250114_np_comedef_avis-sur-l%27usage-des-technologies-d%27intelligence-artificielle-par-les-forces-armees.pdf.

Conclusion

La vision par ordinateur est probablement l'un des domaines les plus matures et les plus opérationnels de l'IA, y compris en matière de défense. Les capacités de détection, segmentation et classification sont clefs pour rendre les systèmes plus autonomes et plus efficaces, et pour optimiser l'allocation de la ressource humaine sur un spectre de tâches et une masse de données qui ne cessent de s'élargir.

Si les techniques classiques de traitement d'images restent pertinentes pour les cas simples et bien contraints, elles sont souvent bien moins performantes que les approches *deep learning* dans des environnements visuels trop spécifiques et des tâches complexes. **Pour cette raison, depuis les années 1990, les chercheurs conçoivent des réseaux de neurones intégrant des processus de convolution. La gamme YOLO en est emblématique, et fournit encore les modèles les plus industrialisables**, grâce à leur frugalité et leur rapidité d'exécution. Toutefois, le paradigme *transformer*, introduit en 2017 pour traiter le langage naturel, trouve aussi son intérêt dans le traitement d'image : en permettant une attention globale et délocalisée, il ouvre la porte à de meilleures interprétations automatiques, au-delà de la simple observation locale. Souvent entraînés de manière auto-supervisée, ces nouvelles approches permettent de s'affranchir de l'annotation exhaustive des données d'entraînement.

En effet, les architectures de réseaux elles-mêmes ne pilotent pas à elles seules la performance des modèles : quantité, qualité et prétraitement données d'entraînement, hyperparamétrage, arbitrages de métriques et autres éléments techniques doivent absolument être choisis avec attention pour aboutir à un modèle performant dans un cas complexe. C'est d'autant plus vrai que **de nouveaux défis techniques s'ajoutent à un travail d'ingénierie déjà complexe** : frugalité, fiabilité, sécurisation des données ou explicabilité sont par exemple encore des axes de recherche.

Complété par une réflexion doctrinale, ce panorama technique permet donc de comprendre pourquoi et comment les modèles de vision se sont imposés comme des incontournables en matière de défense. **Les applications en développement incluent la détection de cibles, la navigation autonome, le support à l'analyse du renseignement, ou encore le sauvetage d'homme à la mer. Elles ne cessent d'être enrichies**, au fil du tempo technologique extrêmement rapide de la recherche en IA. Ainsi, même si, en France, l'homme doit absolument « rester dans la boucle », **il semble clair que la vision par ordinateur s'est rendue indispensable dans les systèmes d'armes et de défense… et le restera longtemps.**






publication@jeunes-ihedn.org