Rapport intermédiaire

# Préambule

Ce rapport intermédiaire contient l’état de l’art des technologies disponibles et utiles à ce travail de Bachelor. Il est divisé en plusieurs sections, chacune traitant un bloc composant l’application.

Pour rappel, nous parlerons ici des différents types de modèle deep learning se prêtant au mieux au contexte énoncé, puis nous réaliserons une synthèse des différents jeux de données trouvés en exposant leurs forces et limitations. Une fois ces deux éléments choisis, nous pourrons statuer sur une architecture pré-entrainée adaptée aux mobiles.

Ensuite, nous nous concentrerons sur la recherche de solutions existantes ou non permettant la réalisation de l’intégration d’un modèle deep learning entrainé à une application cross platform. Dans un premier temps, cette section sera indépendante du modèle deep learning choisi puisque nous sommes, dans l’idéal, à la recherche d’une solution faisant abstraction de la couche machine learning.

Finalement, une analyse architecturale mettra les éléments retenus jusqu’ici en commun et explicitera leurs intégrations dans l’application prototype finale.

# Modèles deep learning

Plusieurs fois au cours de ce rapport, nous avons fait mention de modèle « deep learning ». Notons ici que le terme est générique et ne définit pas un modèle à proprement parlé. Il s’agit plutôt d’une famille d’apprentissages automatiques fondée sur l’apprentissage de représentations de données[[1]](#footnote-1). Dit autrement, le terme définit une technique utilisée par un ensemble de modèles comme l’extraction de caractéristique dans une image ou dans un son. Toutefois, par souci de simplicité, nous utiliserons ce terme pour parler de modèles de machine learning traitant des images au travers d’un réseau neuronal convolutif.

Le réseau de neurones convolutifs (CNN[[2]](#footnote-2)) sont un élément fondamental pour la vision par ordinateur. Depuis son invention en 1988 par Kunihiko Fukushima, il a été largement amélioré et agrémenté de nouvelles possibilités au cours des 12 dernières années avec l’augmentation de puissance des processeurs graphiques (GPU) et la démocratisation de l’intelligence artificielle générative. Ils ont donc la possibilité de répondre à diverses tâches comme la classification d’images, la détection d’objet et la segmentation d’images[[3]](#footnote-3). Nous allons développer ces trois dernières tâches car elles représentent des solutions pertinentes à notre situation.

Une image contenant texte, capture d’écran, diagramme

Description générée automatiquement

Figure 1 - Exemples de type de problèmes de vision par ordinateur[[4]](#footnote-4)

## État de l’art

Certaines techniques ont été volontairement omises dans ce rapport car jugées trop complexes pour les besoins de ce travail. C’est le cas notamment de la segmentation d’image, présentée dans la Figure 1, ou de la détection de points clés (technique ayant pour but d’identifier certains points important d’un objet dans une image, comme les jambes et bras d’une personne).

### Classification d’images

Nous entendons par classification d’image, le processus permettant de catégoriser l’appartenance d’une image à une classe parmi celles d’un ensemble prédéfini. Plus particulièrement dans cette tâche, il s’agit de porter un regard sur l’ensemble de l’image sans spécifier au modèle les régions d’intérêts permettant de prédire la classe correctement. Par conséquent, et comme le démontre la Figure 1, cette technique applique une seule classe par entrée.

#### Avantages

Ce type de modèle constitue une base qui sera présente dans tous les autres que nous traiterons ici. De ce fait, l’architecture de ces modèles seront souvent moins complexes que les autres tâches ce qui pourrait avoir de nombreux avantages bénéfiques sur une plateforme mobile.

Nous pouvons, par exemple, penser à l’espace mémoire moindre occupé par de tels modèles, mais aussi à des temps d’exécution plus courts ou à des besoins en ressources plus faible, ce qui peut s’avérer critique pour des modèles de smartphone d’entrée de gamme.

L’entraînement de tels modèles pourraient également s’avérer plus rapide, bien que pour ce travail, il ne s’agisse pas vraiment d’un avantage à considérer étant donné que l’entraînement sera réalisé sur un ordinateur disposant de d’avantages de ressources.

#### Limitations

Les limitations que présentent un classificateur d’images sont essentiellement liées à la précision de ce dernier. Puisque nous n’indiquons pas la région d’intérêt dans l’image permettant de déterminer sa classe, nous perdons un certain contrôle sur le fonctionnement même du modèle. Ce dernier pourrait, par exemple, déterminer la classe « frelon asiatique » en se basant sur l’arrière-plan de l’image car dans la majorité des cas, les photos de l’insecte ont lieu dans un décors naturel, résultant ainsi par des faux négatifs lorsque l’image reçue présente une scène en intérieur.

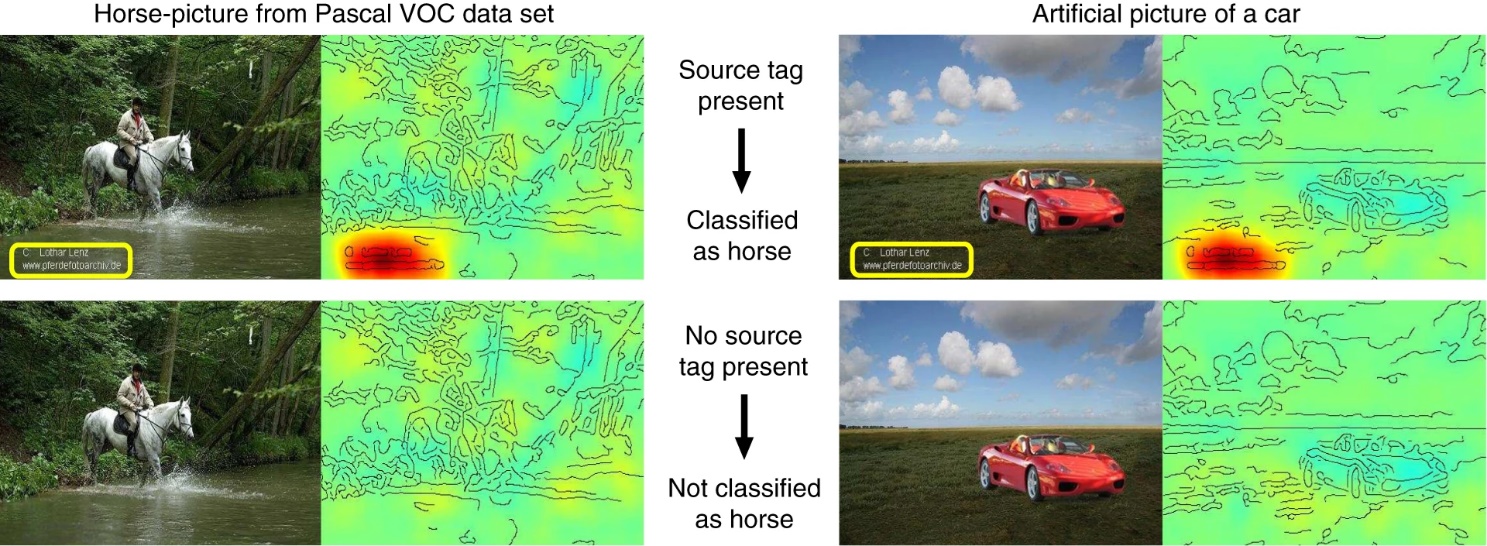


Figure 2 - La présence d'un watermark activant fortement le modèle pour déterminer la classe "cheval" [[5]](#footnote-5)

L’autre défaut que présente les classificateurs sont leur difficulté à identifier correctement une classe si le sujet de l’image est trop petit ou occulté. Il s’agit d’une conséquence directe au fait que le modèle utilise l’image dans sa totalité pour déterminer une classe. Puisque, dans notre cas, nous nous intéressons à des sujets de petites tailles et dont les photographies risquent d’être prise à des points de vue distants, cela pourrait s’avérer être une limitation importante à considérer, d’autant plus si on ajoute à cela les faibles zooms des appareils mobiles et la résolution basse des appareils d’entrée et milieu de gamme.

### Détection d’objets

La détection d’objets est un procédé disposant de nombreux cas d’utilisations comme le pilotage de véhicules autonomes ou la surveillance par vidéo. Comme le montre la Figure 1, il ne s’agit plus de classifier une image dans son ensemble, mais d’identifier une région dans celle-ci et de lui attribuer une classe (voir, dans certains cas, une probabilité d’appartenance à cette dernière).

Ce procédé ouvre la porte à la détection de multiples objets au sein d’une même image, chaque objet pouvant être attribué à une classe distincte.

Par le passé, ce procédé nécessitait l’utilisation d’un CNN pour établir des régions de proposition dans l’image puis un autre réseau pour établir une classification pour chaque région. Ces dernières années, d’autre architectures ont vu le jour permettant d’accélérer ce processus et permettant la sélection de régions et la classification en une seule étape.

#### Avantages

Les résultats des détecteurs d’objets sont plus facilement interprétables une fois l’image analysée par le modèle. Plutôt que d’obtenir une simple classe, nous disposerons également d’une région où se trouve l’élément identifier, sa taille et également la probabilité d’appartenance à la classe. Cela peut s’avérer être avantageux afin de mieux identifier ce que le modèle voit. Cette caractéristique s’avèrerait fortement pratique si on dispose d’un cliché d’une ruche d’abeille attaquée par un frelon. Le modèle pourrait mettre en avant l’insecte dans l’image

Il est plus facile de guider le modèle sur les caractéristiques de l’objet à identifier via le positionnement de l’objet dans l’image. On réduit ainsi l’activation neuronale des zones non pertinentes pour la classification.

Par extension, les détecteurs d’objets apprennent mieux des images où le sujet est occulté, difforme ou de petite taille puisque nous aiguillons la zone à analyser limitant les comportements imprévisibles qui pouvaient naître des classificateurs.

Comme mentionné précédemment, les images à analyser pourront certainement inclure des insectes de petites tailles, en mouvement (donc partiellement flou) ou pris en photo de loin. La détection d’objets pourrait s’avérer plus performante pour identifier un frelon asiatique dans ces scénarios.

La détection d’objet offre la possibilité de classifier plusieurs éléments au sein d’une même image. Ce qui n’est pas possible dans un classificateur.

#### Limitations

Les architectures pré-entrainées de détection d’objet peuvent présenter de grandes différences les unes avec les autres ce qui aura, par conséquence, de fortes variations dans les performances et les ressources nécessaires. Cela est également vrai pour le temps d’apprentissage et d’exécution[[6]](#footnote-6).

Dans la globalité, ces modèles prennent plus d’espace mémoire et consomment d’avantages de ressources qu’un simple classificateur. Ces ressources étant de facto plus faible sur mobile, certaines architectures pourraient mal fonctionner ou ne pas fonctionner du tout. Le temps d’exécution pourrait également s’en trouver augmenté.

La réelle nécessité de détecter la position de l’objet est discutable. Même si elle apporte de précieuses informations quant au fonctionnement du modèle et offre d’avantages de scénarios d’utilisation, les contraintes que le modèle peut imposer à un téléphone mobile peuvent péjorer le choix de cette solution. En plus de cela, la détection d’objets s’utilise d’avantage dans des contextes vidéos où il y a une nécessité de suivre la position d’un objet au cours du temps.

## Point de situation sur la recherche

Avant d’orienter notre choix vers une solution adaptée à notre problématique, il est important de souligner le fait que la recherche actuelle sur l’intégration de modèles deep learning sur téléphones mobiles n’est pas un sujet traité de façon exhaustive. Les recherches que nous avons trouvées se concentrent bien trop souvent sur un nombre d’architecture limité, ou en exécutant les modèles sur un faible nombre de smartphone voir dans certains cas en ignorant complètement l’aspect cross-platform en excluant volontairement certains OS.

Il en découle des résultats variés et variables exposant des métriques différentes. Certaines études mettent en lumière les coûts en termes de temps et de consommation de batterie, alors que d’autres mettent en avant la précision du modèle et le nombre de paramètre de ce dernier.

Ainsi, les observations jusqu’ici ont été faites sur un ensemble faible d’études. Toutefois, puisque l’objectif principal de ce travail s’axe plutôt sur la faisabilité de l’intégration du modèle deep learning sur un smartphone, les informations récoltées sont suffisantes pour nous orienter sur un choix éclairé.

## Solution choisie

Les deux techniques présentées ci-dessus nous ont semblé être les plus pertinentes et adaptées pour ce projet. Nous avons décidé d’orienter notre choix final sur une technique de classification d’image.

En effet, les contraintes de ce modèle peuvent être aisément contournée pour les besoins de ce travail. Tout d’abord, nous n’avons pas réellement besoin d’identifier plusieurs éléments au sein d’une même image. À termes, l’application enverrait dans tous les cas le cliché à un serveur ce qui permettra à un œil humain d’identifier plusieurs individus sur l’image si besoin.

En second temps, la problématique liée au sujet de l’image qui serait trop petit peut être contournée si on invite l’utilisateur à recadrer son cliché en ne sélectionnant que la zone contenant l’insecte à identifier. Cela permet non seulement aux utilisateurs de saisir le cliché depuis un point de vue éloigné à des fins sécuritaires tout en obtenant une image avec un sujet mieux centré et au final plus facile à reconnaître.

Les classificateurs d’images offrent une solution simple à entraîner et à utiliser et s’avèrent donc être de bon candidats pour tester leur portabilité dans un téléphone. Les autres avantages qu’ils ont à offrir dans notre contexte ont d’ores et déjà été explicité plus haut.

# Dataset

La section précédente nous a fait nous orienter sur une tâche de classification d’images. De ce fait, nous devons désormais rechercher un jeu de donnée contenant des images du frelon asiatique annoté pour entrainer notre modèle.

le *Vespa Velutina* étant une espèce invasive dans plusieurs pays dont notamment la France et l’Espagne, cette espèce a déjà été le sujet d’observations et de nombreux clichés divers et variés sont trouvable sur internet. L’idéal étant de disposer d’un ensemble de cliché annoté correctement, ce qui est chose possible au travers de diverses plateformes web mettant à disposition des datasets en open source comme *Hugging Face*[[7]](#footnote-7) ou *Robotflow* [[8]](#footnote-8).

# Architectures de Réseau de neurones

Complètement dépendant du point 1.

# Intégration de modèle dans une application cross-platform

TODO

# Modélisation et architecture de l’application

TODO

# Solution choisie

1. Apprentissage profond (2024, 2 juillet). In *Wikipedia.* https://fr.wikipedia.org/wiki/Apprentissage\_profond [↑](#footnote-ref-1)
2. Convolutional neural networks [↑](#footnote-ref-2)
3. Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., Ghayvat, H., (2021). *CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope.* MDPI. https://www.mdpi.com/2079-9292/10/20/2470 [↑](#footnote-ref-3)
4. Mrinal Walia. (Sep 28, 2022). *Object Detection vs. Image Classification vs. Keypoint Detection*. Roboflow Blog: https://blog.roboflow.com/object-detection-vs-image-classification-vs-keypoint-detection/ [↑](#footnote-ref-4)
5. Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K-R., (2019, 11 mars), Unmasking Clever Hans predictors and assessing what machines really learn, *Nature Communications,* *10*. https://www.nature.com/articles/s41467-019-08987-4 [↑](#footnote-ref-5)
6. Rasyad, M. A., Dewanta, F., Astuti, S., (2021), *All-in-one computation vs. computational-offloading approaches: a performance evaluation of object detection strategies on android mobile devices. https://www.researchgate.net/publication/358725019\_All-in-one\_computation\_vs\_computational-offloading\_approaches\_a\_performance\_evaluation\_of\_object\_detection\_strategies\_on\_android\_mobile\_devices* [↑](#footnote-ref-6)
7. https://huggingface.co/ [↑](#footnote-ref-7)
8. https://universe.roboflow.com/ [↑](#footnote-ref-8)