Test Scientifique - Expertise Computer Vision - Quantmetry

Lionel Cheng

Juin 2022

1 Contexte et présentation du problème

L'objectif de ce travail est de proposer une méthodologie et de répondre à des questions autour de la détection automatique en temps réel de l'occupation de places de parking par traitement d'images. La méthode doit pouvoir être implémentable sur un nano-ordinateur de type Raspberry Pi. Deux étapes sont nécessaires pour répondre à ce problème:

- La détection des places de parking à partir de la vidéo d'une caméra
- L'occupation de chaque place

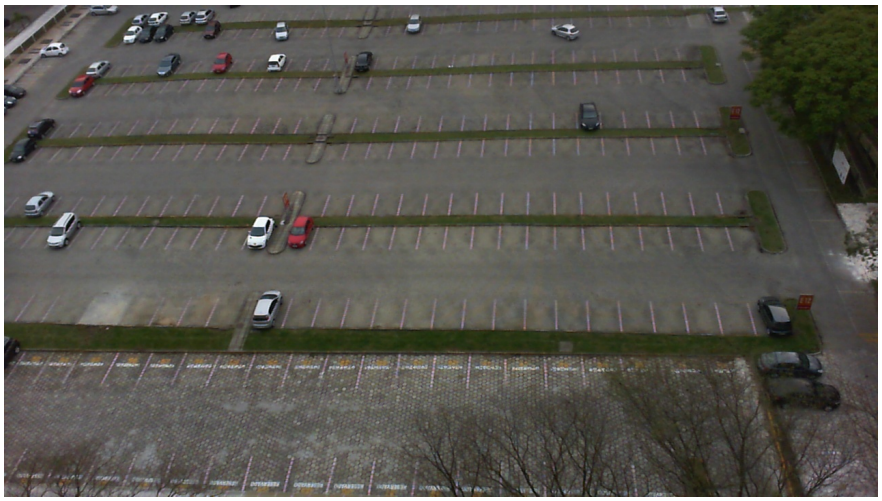
Ce document se focalise sur la deuxième étape de ce problème, la première pouvant être traitée par un Mask R-CNN [He et al., 2017] ou alors en segmentant directement la caméra que l'on utilise en délimitant les places, les caméras de surveillance de parking étant immobiles la plupart du temps.

Nous avons donc affaire à une tâche de classification d'images à 2 classes: occupée ou libre. Des réseaux de neurones vont être utilisés pour répondre à ce problème, ceux-ci étant particulièrement rapides et efficaces pour le traitment d'images depuis l'avènement d'AlexNet [Krizhevsky et al., 2012].

2 Données

Le jeu de données suggéré dans l'énoncé, PKLot, a été publié en 2015 [de Almeida et al., 2015] et est le premier jeu de données aussi conséquent d'images de parkings. Celui-ci contient 12416 images de parkings provenant de 3 cameras de surveillance (PUC, UFPR04 et UFPR05) prises à différents moments de la journée sur plusieurs jours. Ces images sont segmentées en 695 899 images de places de parking individuelles [de Almeida et al., 2015]. Celles-ci ont des orientations et des éclairages différents en fonction de la météo. On peut voir des images complètes des 3 caméras en Fig. 1 et des images segmentées en Fig. 2.

Considérez-vous que la quantité de données à disposition est suffisante pour entraîner un algorithme d'apprentissage supervisé ?

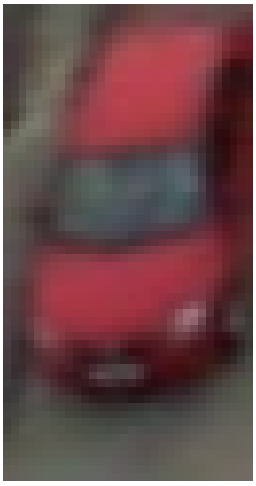


(a) Caméra PUC

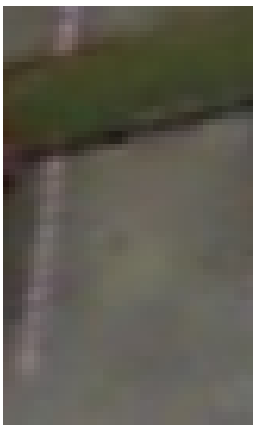
(b) Caméra UFPR04

(c) Caméra UFPR05

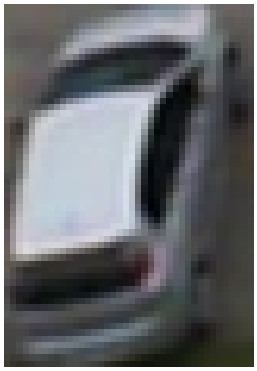
Figure 1: Vue complète des caméras de surveillance du jeu de données PKLot.



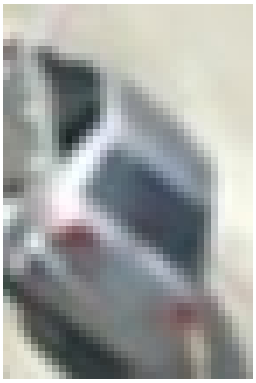
(a) Occupée



(b) Libre



(c) Occupée



(d) Occupée

Figure 2: Images de places de parkings labellisées provenant du jeu de données PKLot.

Il y a environ 350 000 images par classe dans le jeu de données PKLot. Ce nombre semble largement suffisant lorsqu'on le compare à d'autres jeux de données d'images classiques tels que ImageNet LSVRC-2010 [Russakovsky et al., 2015] (1200 images par classe en moyenne), MNIST [Deng, 2012] (700 images par classe), CIFAR-10 et CIFAR-100 [Krizhevsky, 2009] (6000 et 600 images par classe respectivement).

Plus que la taille du jeu de données, sa qualité doit être un critère primordial pour entraîner un réseau de neurones. Si celui-ci est trop biaisé dans un sens (une seule position de voiture par exemple) ou si les exemples sont trop simples, le réseau aura du mal à généraliser ses résultats.

Quelle(s) solution(s) proposez-vous pour augmenter la taille du dataset, si nécessaire ?

Lorsque la taille du dataset n'est pas assez conséquente plusieurs techniques s'offrent à nous pour faire de **l'augmentation de données**. Ces opérations permettent d'éviter le sur-apprentissage (ou *over-fitting* en anglais).

Une manière évidente est de réaliser des opérations sur le dataset existant tels que tourner, flouter, redimensionner ou encore contraster les images. Ces opérations sont illustrés en Fig. 3.

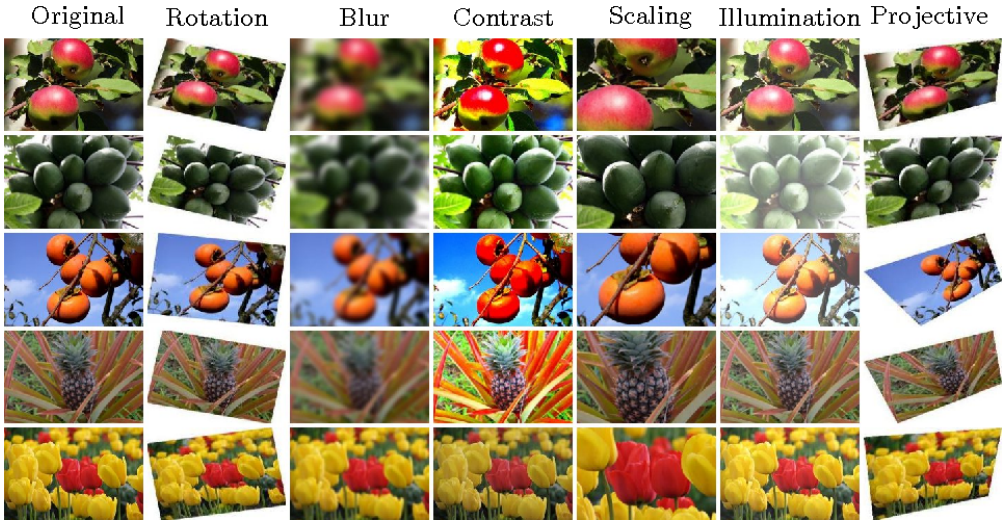


Figure 3: Méthodes d'augmentation de données par modification du dataset existant.

Des techniques plus avancées peuvent être utilisées tels que la génération d'images par GAN (Generative Adverserial Network) comme par exemple en utilisant le CycleGAN de Zhu et al. [2017]. Des illustrations des capacités du CycleGAN sont montrées en Fig. 4.

3 Modélisation

Quel modèle proposeriez-vous ? Quelle stratégie d'entraînement envisageriez vous ? Pourquoi ?

Face à cette tâche de classification d'images, nous nous orientons vers des réseaux de neurones profonds convolutionnels (*deep CNN* en anglais pour Convolutional Neural Network). Ces méthodes sont en effet les meilleurs pour les tâches de classification d'images

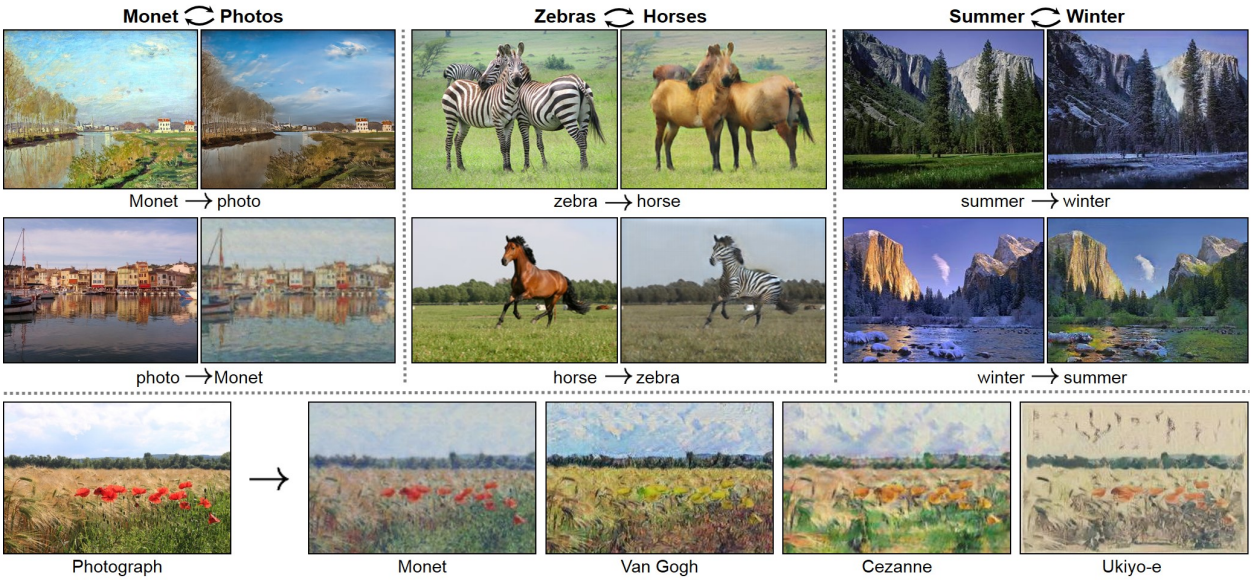


Figure 4: Méthodes d'augmentation de données par génération d'images synthétiques en utilisant le CycleGAN [Zhu et al., 2017].

depuis l'avènement d'AlexNet [Krizhevsky et al., 2012], le premier réseau de neurones à avoir remporté avec beaucoup de marge la compétition ImageNet Large Scale Visual Recognition Challenge en 2012, mentionné en première partie de ce document. L'architecture du réseau est montrée en Fig. 5.

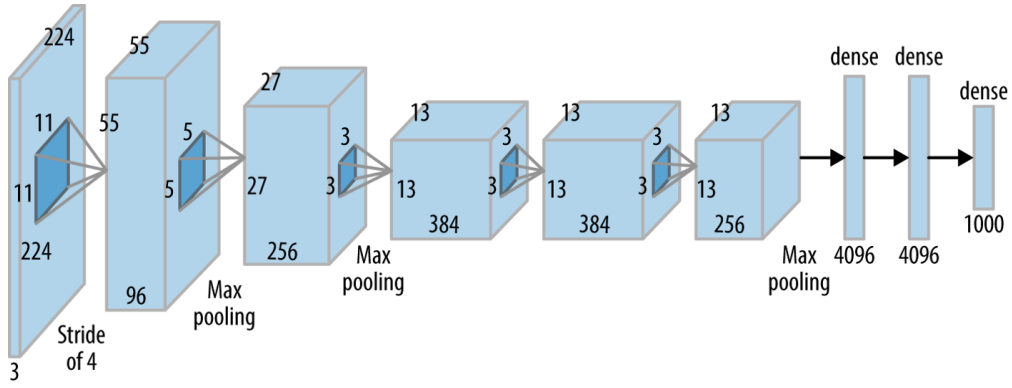


Figure 5: L'architecture d'AlexNet [Krizhevsky et al., 2012].

AlexNet contient environ 60 million de paramètres pour classifier 1000 différents familles d'images. Par rapport à notre tâche de classification binaire, on peut penser que son architecture est bien trop volumineuse et qu'un réseau plus petit sera à même d'être performant pour notre tâche de classification de places occupées ou libres. C'est justement ce qui est fait dans [Amato et al., 2017] où une version allégée d'AlexNet qui contient plus de 1000 fois moins de paramètres est crée sous le nom de mAlexNet. L'architecture de ce réseau réduit est montrée en Fig. 6.

En terms de CNNs plus légers et adaptés à notre cas on peut penser aux CNNs travaillant

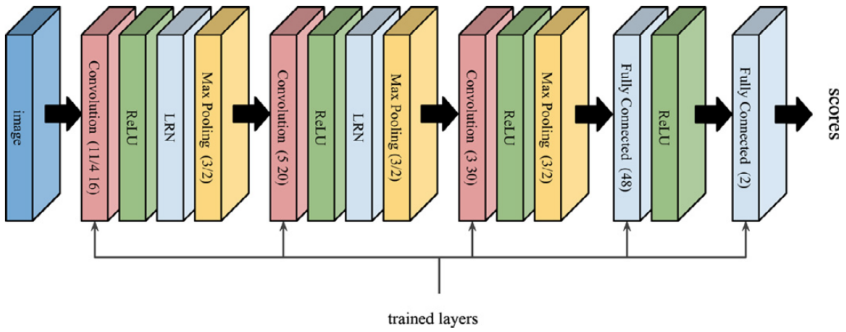


Figure 6: L'architecture de mAlexNet [Amato et al., 2017].

sur le jeu de données MNIST, des images de chiffres en noir et blanc écrits à la main, qui est une tâche de classification comprenant 10 familles (les chiffres de 0 à 9). Le réseau le plus établi pour résoudre cette tâche est le CNN LeNet5 [Lecun et al., 1998] dont l'architecture est montrée en Fig. 7.

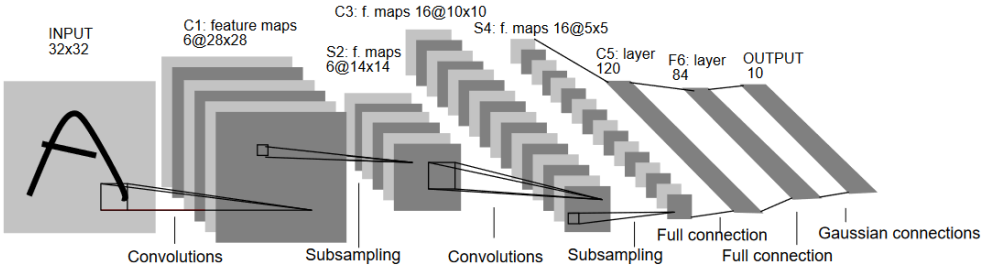


Figure 7: L'architecture de LeNet5 [Lecun et al., 1998].

Tous ces réseaux suivent une logique similaire: en partant de l'image pixellisée (avec un champ de couleur pour MNIST ou 3 pour les images de parking ou ImageNet), plusieurs couches de convolution/pooling sont appliquées avec des fonctions d'activation puis ces couches de convolution sont mises à plat (*flatten* en anglais) et des couches de perceptrons (*multi-layer perceptron* en anglais) sont ensuite appliquées. La dernière couche comprend autant de perceptrons que le nombre de classes et une opération de Softmax qui convertit ces amplitudes en probabilités est réalisée. La fonction de coût utilisée correspond alors à une entropie croisée (*cross-entropy* en anglais):

$$l(x, y) = \frac{1}{N} \sum_{n=1}^{N} l_n \quad \text{où} \quad l_n = -\log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \tag{1}$$

où N représente le batch-size et $x_{n,c}$ correspond à la dernière couche du réseau avant l'opération de Softmax qui est explicitée dans la fonction de coût ici.

Quelles métriques de performance / stratégies de validation sont envisageables pour évaluer le comportement de votre modèle ?

Dans cette tâche de classification binaire, la métrique de performance la plus évident est l'exactitude (ou *accuracy* que l'on dénote ACC) en anglais qui est la proportion de places

correctemment classifiées sur le nombre de places total. Pour définir d'autres métriques et comprendre un peu plus le comportement du réseau on fait appel à la matrice de confusion qui permet d'utiliser des grandeurs conditionnées. Celle-ci est montrée en anglais en Fig. 8 pour garder des notations consistentes avec celles du code. Les images sont classées en 4 catégories: le nombre de vrais positif, vrais négatifs, faux positifs et faux négatifs. On considère ici le test positif lorsque la place est occupée. L'exactitude ACC s'écrit alors:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{TN} + \mathrm{FP}} \tag{2}$$

On définit de plus les métriques conditionnées suivantes:

- Le taux de vrais positifs ou *true positive rate* (TPR) en anglais:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3}$$

- Le taux de vrais négatifs ou *true negative rate* (TNR) en anglais:

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{4}$$

- La valeur prédictive positive ou *positive predictive value* (PPV) en anglais:

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{5}$$

- La valeur prédictive negative ou *negative predictive value* (NPV) en anglais:

$$\mathrm{NPV} = \frac{\mathrm{TP}}{\mathrm{TN} + \mathrm{FN}} \tag{6}$$

Afin de déterminer le meilleur modèle, on utilise pendant l'entraînement un jeu de données de validation (tiré du jeu d'entraînement à une proportion 80/20 si il n'y en a pas de disponible explicitement). Le meilleur modèle sera considéré lorsque l'exactitude sur le jeu de donneés de validation sera maximale au cours de l'entraînement. L'évaluation finale du réseau ainsi entraînée peut enfin être réalisée sur des jeux de données dits *tests* que le réseau n'a jamais vu au cours de l'entraînement.

Résultats

Une librairie appelée `pklotclass` a été créée pour entraîner une version adaptée de mAlexNet, AlexNet et LeNet5. Celle-ci se base sur la librairie PyTorch [Paszke et al., 2019] pour tester les différents architectures. Pour réaliser des tests rapides, on se base sur le jeu de données CNRPark [Amato et al., 2017] qui contient environ 12 000 images et qui est séparé en CNRParkEven (6413 images) et CNRParkOdd (6171 images). Les instructions pour reproduire les résultats mentionnés ici sont disponibles dans le `README.md` du répertoire github fourni avec ce document.

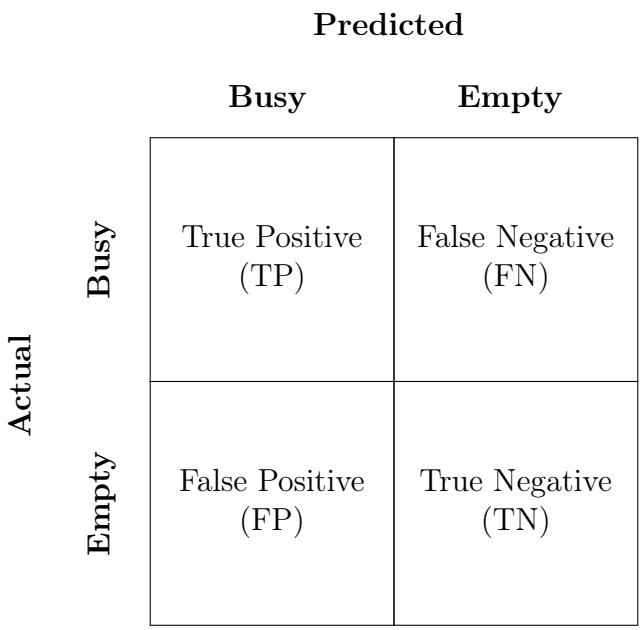


Figure 8: Matrice de confusion pour notre tâche de classification.

Les trois réseaux contiennent respectivement 14 585 538, 5 406 650 et 32 380 paramètres pour AlexNet, LeNet5 et mAlexNet. Le jeu de données CNRParkEven a été divisé en parts de 80/20 pour l'entraînement et la validation. On utilise un optimiseur de descente stochastique de gradient (SGD en anglais) dont les paramètres sont tirés de ceux utilisés dans Amato et al. [2017] avec un planificateur qui réduit le taux d'apprentissage de moitié lorsque l'exactitude de validation n'augmente pas deux fois.

L'entraînement est arrêté à 18 époques et les courbes de fonction de coût et exactitude sont montrées en Fig. 9 pour le jeu de données de validation CNRParkOdd. On voit que LeNet5 est meilleur que les deux architectures de type AlexNet que ce soit en termes de fonction de coût et exactitude. Comparant AlexNet et mAlexNet, le premier arrive à obtenir de meilleur résultats que le second mais possède environ 1 000 fois plus de paramètres. Le fait de pouvoir atteindre une exactitude dépassant les 90% avec un réseau aussi petit que mAlexNet est néanmoins très encourageant.

Lorsque l'on regarde les statistiques plus avancées en Fig. 10 (une valeur de -1 correspond à un dénominateur nul dans les formules), on peut noter encore une fois la proximité de comportement d'AlexNet et mAlexNet et les meilleurs résultats de LeNet5. On remarque globalement que les métriques concernant les places vides (TNR et NPV) sont toujours inférieures à leurs contreparties "places pleines" (TPR et PPV). Il semble donc plus difficile pour le réseau de prédire qu'une image de place de parking est vide sachant que la place est vide (TNR) que de prédire qu'une image de parking est occupée sachant qu'elle est occupée (TPR).

L'exactitude ainsi que les différents métriques avancées des trois réseaux pour les paramètres produisant la meilleure exactitude sur les données de validation sont montrées en Tab. 1. On peut remarquer qu'AlexNet et LeNet5 ont une exactitude similaire (91.6% et 91.5% respec-

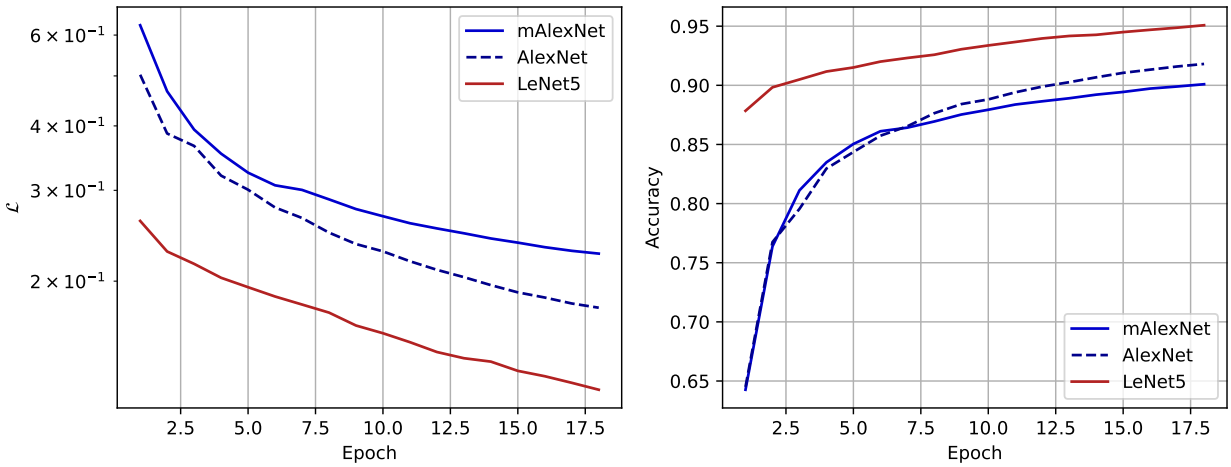


Figure 9: Loss et Accuracy d'AlexNet, mAlexNet and LeNet5 entraînées sur CNRParkEven sur le jeu de données CNRParkOdd.

tivement). Les métriques conditionnées révèlent cependant des différences de comportement: le taux de vrais positifs d'AlexNet est supérieure à celui de LeNet5 tandis que le taux de vrais négatifs de LeNet5 est supérieure à celui d'AlexNet.

Network	ACC	TPR	TNR	PPV	NPV
mAlexNet	0.897	0.949	0.780	0.906	0.873
AlexNet	0.916	0.960	0.818	0.922	0.902
LeNet5	0.915	0.940	0.857	0.936	0.865

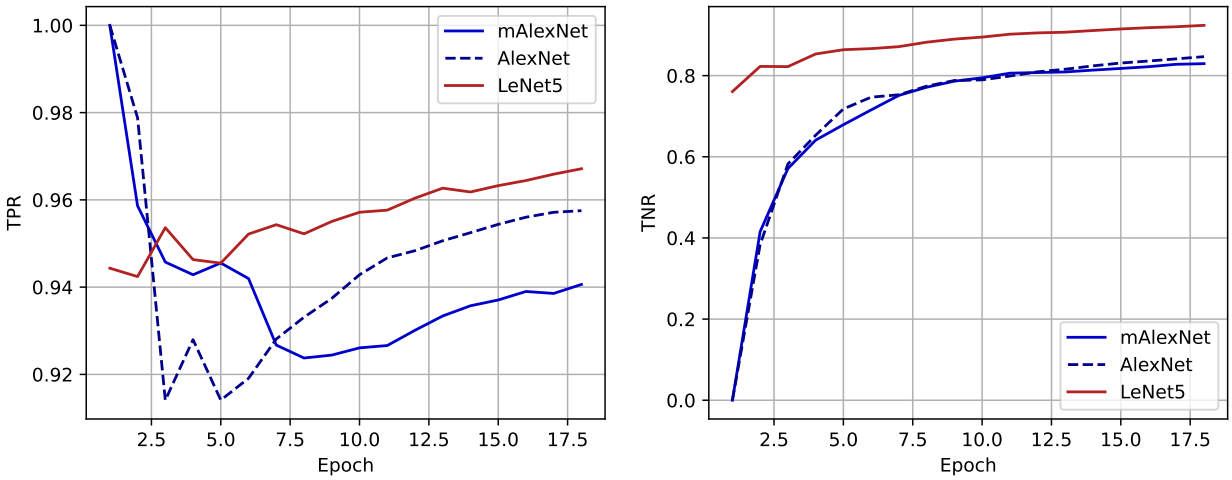
Table 1: Métriques sur le jeu de données de test CNRParkOdd d'AlexNet, mAlexNet et LeNet5

4 Optimisation

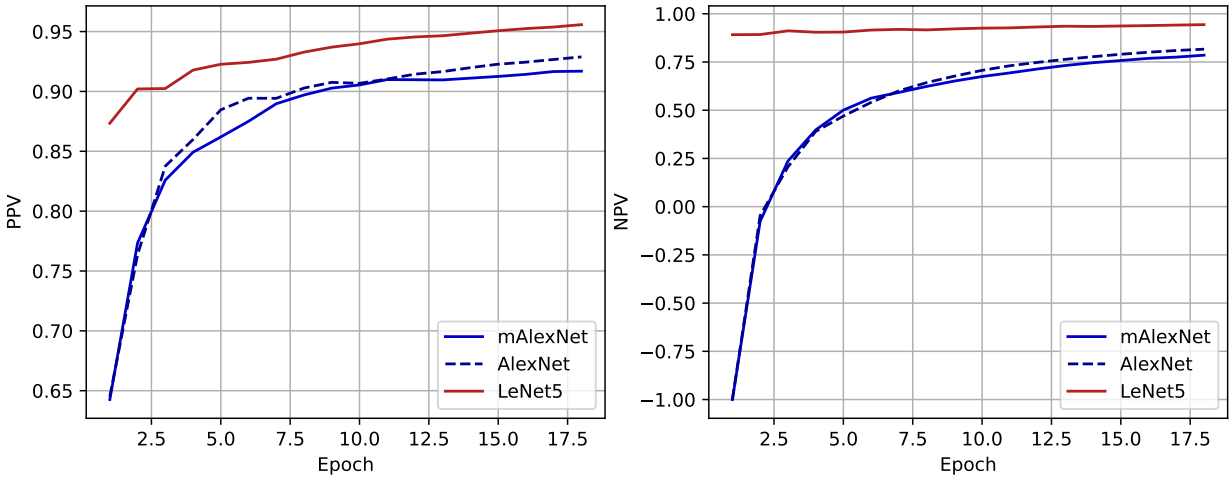
Une fois le modèle développé et mis en production, on constate que ses performances sont moins bonnes que lors des tests effectués pendant sa phase de développement. Le principal facteur identifié est que les voitures blanches ne sont pas (ou mal) détectées lorsque le parking est recouvert de neige.

Quelle pourrait être la cause de ce défaut ? Proposez une démarche permettant de le résoudre et d'augmenter la performance du modèle dans ce type de situations.

La principale cause de ce défaut est l'absence d'images similaires dans le jeu de données d'entraînement qui n'est pas assez représentatif de la réalité. C'est la raison pour laquelle un nouveau jeu de données CNRPark-EXT est introduit dans le papier [Amato et al., 2017] car le jeu de données PKLot [de Almeida et al., 2015] manque d'images de situations difficiles qui peuvent être rencontrées en réalité comme des occlusions et ombrages montrées en Fig. 11.



(a) TPR et TNR



(b) PPV et NPV

Figure 10: Métriques plus avancées d'AlexNet, mAlexNet and LeNet5 entraînées sur CNRParkEven sur le jeu de données CNRParkOdd.

Il faut donc rajouter dans le jeu de données d'entraînement des images de places de parking enneigées.

Figure 11: Des images du jeu de données CNRPark-EXT provenant de Amato et al. [2017].

Cette question soulève le problème de la représentativité des données d'entraînement et de leur qualité pour entraîner un réseau de neurones. Dans Amato et al. [2017], la comparaison des performances des mêmes réseaux entraînés et testés sur différents jeu de données est reproduite en Fig. 12. Bien que le jeu de données CNRPark est petit (12 000 images) par rapport à PKLot TRAIN (plus de 40 000 images), l'exactitude sur le jeu de données CNRPark-EXT TEST est meilleure en entraînant sur CNRPark pour AlexNet et mAlexNet. Cela montre qu'il faut être vraiment soucieux de la représentativité et de la qualité du jeu de données d'entraînement.

5 Interprétabilité

Quantmetry a rejoint le collectif français Confiance.ai lancé par l'Etat dans le cadre du Grand Défi "Sécuriser, certifier et fiabiliser les systèmes fondés sur l'intelligence artificielle". A ce titre, l'interprétabilité des modèles est une problématique de l'IA (et a fortiori de la Computer Vision) que Quantmetry propose de résoudre pour ses clients.

Quelle(s) solution(s) proposeriez-vous pour interpréter vos résultats ? pour ajouter une évaluation de l'incertitude des prédictions de l'algorithme développé ? Quels seraient leurs avantages et limites ?

Bien que les réseaux de neurones profonds ont montré des résultats impressionants sur de nombreuses tâches, leur interprétabilité est complexe en raison du très grand nombre de paramètres en jeu. Dans de nombreux cas, ces réseaux sont utilisés comme des boîtes noires ce qui peut engendrer beaucoup de risque. L'interprétabilité n'est pas un concept monolithique et englobe plusieurs aspects définis dans Chakraborty et al. [2017]:

- La transparence du modèle: cet aspect comprend la reproductibilité du modèle, sa décomposabilité (est ce qu'il y a une explication intuitive des paramètres ?) et enfin sa transparence algorithmique (une manière d'expliquer comment il apprend)

Table 3
Experiments performed to test the generalization performance of *mAlexNet* and *AlexNet*. Accuracies on test sets are reported, for each combination of (model, training set, test set).

Method	Test set	Accuracy	AUC
TRAIN ON CNRPARK			
mAlexNet	CNRPark-EXT TEST	93.52%	0.9838
AlexNet	CNRPark-EXT TEST	93.63%	0.9877
mAlexNet	PKLot TEST	95.28%	0.9916
AlexNet	PKLot TEST	95.60%	0.9910
TRAIN ON CNRPARK+EXT TRAIN C1-C8			
mAlexNet	CNRPark-EXT TEST	95.88%	0.9937
AlexNet	CNRPark-EXT TEST	96.85%	0.9957
mAlexNet	PKLot TEST	90.48%	0.9738
AlexNet	PKLot TEST	96.51%	0.9937
TRAIN ON CNRPARK+EXT TRAIN			
mAlexNet	CNRPark-EXT TEST	97.71%	0.9967
AlexNet	CNRPark-EXT TEST	98.00%	0.9974
mAlexNet	PKLot TEST	84.53%	0.9699
AlexNet	PKLot TEST	93.70%	0.9923
TRAIN ON PKLOT TRAIN			
mAlexNet	PKLot TEST	98.07%	0.9967
AlexNet	PKLot TEST	98.81%	0.9984
mAlexNet	CNRPark-EXT TEST	83.83%	0.9139
AlexNet	CNRPark-EXT TEST	90.52%	0.9684
TRAIN ON PKLOT2DAYS			
mAlexNet	CNRPark	82.88%	0.899
LBP*	CNRPark	65.31 %	0.580
TRAIN ON CNRPARK			
mAlexNet	PKLot	90.38%	0.989
LBP*	PKLot	52.88 %	0.391

* de Almeida et al. (2015).

Figure 12: Résultats d'entraînement de différents CNNs provenant de Amato et al. [2017].

- La fonctionnalité du modèle: la sortie du modèle est-elle explicable en quelques phrases ? Peut-on visualiser simplement ses paramètres ? Quels sont les paramètres critiques pour la variabilité du modèle ?

Ces dernières années, de nombreuses méthodes pour expliquer ce qui se passe à l'intérieur d'un CNN ont été développé Zhang and Zhu [2018]. Plusieurs approches sont ainsi possibles:

- Visualiser les représentations intermédiaires du CNN à travers ses couches intermédiaires et les poids des kernels de convolution
- Diagnostic des représentations: trouver une explication sémantique à chaque couche utilisée ou encore extraire des zones d'images qui contributent le plus à la sortie du réseau
- Désintriquer les représentations de CNNs en créant des représentations en graphes des zones que le réseau "regarde".

6 Frugalité

Quelles démarches proposeriez-vous pour optimiser les besoins en ressources de votre solution ?

La démarche proposée dans Amato et al. [2017] d'utiliser un réseau adapté à la complexité du problème donne des pistes pour optimiser les besoins en ressources d'un CNN par rapport à un problème: réduire d'un facteur 1000 le nombre de paramètres n'a pas changé significativement la performance du réseau pour prédire l'occupation des places de parking mais l'empreinte mémoire et la rapidité du réseau ont été grandement augmentés. On pourrait éventuellement pousser la réduction du nombre de paramètres pour voir son impact sur l'exactitude de la prédiction.

Si le réseau que l'on utilise est fixé, il faut regarder si certaines parties du code sont sous-optimisés et prennent beaucoup de temps: pour cela profiler le code est nécessaire pour savoir où sont les parties à améliorer.

References

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. URL https://arxiv.org/abs/1703.06870.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Paulo R.L. de Almeida, Luiz S. Oliveira, Alceu S. Britto, Eunelson J. Silva, and Alessandro L. Koerich. PKLot – a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937–4949, July 2015. doi: 10.1016/j.eswa.2015.02.009. URL https://doi.org/10.1016/j.eswa.2015.02.009.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, Carlo Meghini, and Claudio Vairo. Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, 72:327–334, April 2017. doi: 10.1016/j.eswa.2016.10.055. URL https://doi.org/10.1016/j.eswa.2016.10.055.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M. Rao, Troy D. Kelley, Dave Braines, Murat Sensoy, Christopher J. Willis, and Prudhvi Gurram. Interpretability of deep learning models: A survey of results. In *2017 IEEE Smart-World, Ubiquitous Intelligence and Computing, Advanced and Trusted Computed, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, August 2017. doi: 10.1109/uic-atc.2017.8397411. URL https://doi.org/10.1109/uic-atc.2017.8397411.

Quanshi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey, 2018. URL https://arxiv.org/abs/1802.00614.