MobileFlow : modèle et mise en œuvre pour une inférence de flot optique efficace

Mickael Seznec, Agathe Archet, Nicolas Gac, François Orieux, Alvin Sashala Naik

Mobileflow : modèle et mise en œuvre pour une inférence de flot optique efficace

Mickaël SEZNEC[1,2], Agathe ARCHET[1,2] Nicolas GAC[2], François ORIEUX[2], Alvin SASHALA NAIK[1]

[1]Thales Research and Technology,
1 Avenue Augustin Fresnel, 91120 Palaiseau, France

[2]Université Paris-Saclay, CNRS, CentraleSupélec,
Laboratoire des Signaux et Systèmes, 3 rue Joliot Curie, 91190 Gif-Sur-Yvette, France

mseznec@nvidia.com, agathe.archet@l2s.centralesupelec.fr
nicolas.gac@l2s.centralesupelec.fr, orieux@l2s.centralesupelec.fr
alvin.sashalanaik@thalesgroup.com

Résumé – Estimer le flot optique entre deux images est une application particulièrement coûteuse en ressources de calcul et de temps. Cela pose notamment problème pour des déploiements temps-réel sur des plateformes embarquées, où les solutions algorithmiques doivent être légères, rapides et rester performantes. Ainsi, nous proposons MobileFlow, un réseau de neurones convolutif allégé, basé sur PWC-Net et MobileNetV2. Ce réseau est plus performant (- 12% sur l'EPE, ou l'EndPoint Error), plus compact (-91% de poids de paramètres) et plus rapide (+ 14% FPS en précision fp32) que PWC-Net sur le dataset Flying Chairs.

Abstract – Estimating the optical flow between two images is a particularly costly application with regard to computational and time resources. This is particularly problematic for real-time deployments on embedded platforms, where algorithmic solutions must be light, fast and remain efficient. Thus, we propose MobileFlow, a lightweight convolutional neural network, based on PWC-Net and MobileNetV2. This network is more efficient (- 12% on EPE, or EndPoint Error), more compact (-91% on parameters' weight) and faster (+ 14% FPS with fp32 precision) than PWC-Net on Flying Chairs dataset.

1 Introduction

On désigne par flot optique le mouvement apparent des pixels entre deux observations temporelles successives d'une même scène. Les informations extraites peuvent servir de base pour d'autres applications de traitement d'images, comme l'analyse de scène par exemple. Les approches traditionnelles reposent sur la minimisation d'une énergie [1]. Elles proposent des solutions robustes aux problèmes d'occultation, d'illuminations ou de bruit, mais restent toutefois coûteuses en temps de calcul. En réponse à ce premier obstacle, l'emploi de réseau de neurones convolutifs (CNN) offre une solution alternative à la fois plus fiable, avec une meilleure efficacité de calcul et un temps d'inférence réduit [2, 3]).

Aujourd'hui, les nombreuses applications menées sur des systèmes autonomes imposent aux solutions algorithmiques de s'adapter également aux contraintes de l'embarqué et du temps réel. Dans ce contexte, l'espace, la consommation énergétique, la puissance de calcul réduites et le temps deviennent très limités. Ainsi, les nouveaux besoins demandent des algorithmes fiables, toujours plus rapides et légers. Pour y parvenir, la tendance actuelle se focalise sur la conception de réseau de neurones plus compacts. Cela, grâce à des réseaux aux architectures plus efficaces, par exemple avec les MobileNets [4, 5, 6] et SqueezeNet [7], ou par la mise en place de stratégies d'optimisation post-entraînement, comme le pruning [8] et la quantification [9].

Dans cette démarche, nous proposons la conception et l'implémentation de MobileFlow, un modèle CNN compact pour le flot optique sur une cible embarquée. Nos contributions sont les suivantes :

- La conception de MobileFlow, un réseau léger et efficace pour l'estimation de flot optique basé sur PWC-Net [3],
- L'utilisation de MobileNetV2 comme extracteur multi-échelle des caractéristiques des images de PWC-Net, allégeant ainsi le réseau,
- Un retour d'expérience du cheminement partant de l'entrainement du réseau avec un framework haut-niveau (Pytorch) jusqu'au déploiement sur une cible embarquée.
- Un profiling complémentaire pour analyser les voies d'accélération possibles pour MobileFlow.

2 Architecture de MobileFlow

L'architecture de PWC-Net constitue le point de départ pour notre réseau. Réutilisé pour d'autres applications, PWC-Net a démontré son efficacité dans le cas du flot optique et reste relativement compact avec un temps d'inférence raisonnable [3].

Ce modèle opère sur deux images RGB pour générer un flot optique. Un premier double sous-réseau CNN classique sert d'extracteur et fournit les caractéristiques de chacune des images à différentes échelles de plus en plus réduites. Un second sous-réseau estime et affine le flot optique en cascade, en exploitant la pyramide de caractéristiques précédemment obtenue. Cette partie aborde les choix d'optimisation sur le modèle, indépendants de la cible matérielle.

2.1 Extraction des caractéristiques

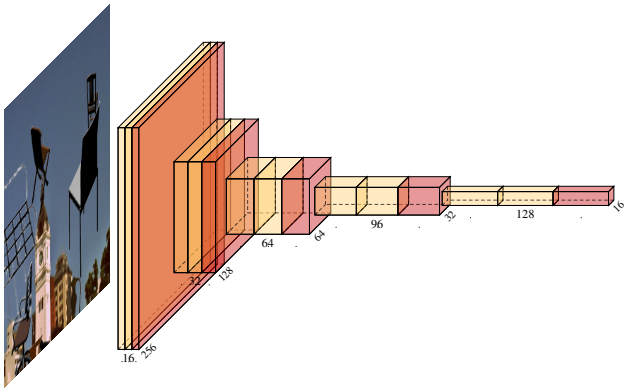

FIGURE 1 – Les cinq premiers niveaux de couches du sous-réseau extracteur de PWC-Net. Les caractéristiques issues des couches en rouge servent à l'estimation du flot en cascade.

Nous choisissons de remplacer le premier sous-réseau extracteur de PWC-Net (Figure 1) par MobileNetV2 [5] pour 3 raisons :

— MobileNetV2 est un réseau équivalent plus léger. Ses couches de convolution séparables sont plus économes en paramètres et en opérations que les couches de convolutions classiques, avec une performance presque préservée.

— MobileNetV2 a déjà joué le rôle d'extracteur dans d'autres applications. Combiné à SDD, MobileNetV2-SDD s'avère efficace dans le cas de la détection d'objets, bien que MobileNetV2 soit initialement entraîné pour de la classification.

— La réutilisation des poids pré-entraînés de MobileNetV2 est possible. Le *transfer learning* permet un gain de temps sur l'entraînement, et peut aider à une meilleure généralisation du modèle pour les poids concernés, entrainés alors sur des tâches de classification et de flot optique.

Comme initialement prévu avec MobileNetV2, un paramètre α permet d'ajuster la profondeur (*channels*) des couches de convolution du réseau extracteur.

2.2 Estimation du flot optique

Une fois les caractéristiques multi-échelles extraites depuis les deux images, PWC-Net cherche ensuite à estimer le flot optique pour chaque niveau de la pyramide, comme détaillé en Figure 2.

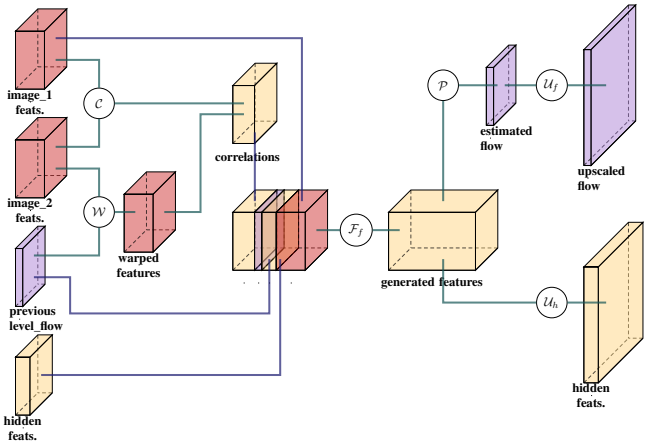

FIGURE 2 – Détail de l'estimateur du flot de PWC-Net pour un niveau. Les cercles représentent les sous-modules, les couches violettes des copies de tenseurs.

Les caractéristiques de la deuxième image subissent une opération de *backwarp* $\mathcal{W}$ en utilisant le flot estimé au niveau précédent, ceci en vue de faciliter l'étape de mise en corrélation suivante. L'étape de corrélation $\mathcal{C}$ calcule les similarités spatiales des deux images. Les données corrélées sont ensuite concaténées à d'autres tenseurs : les caractéristiques de la première image, le flot estimé et des caractéristiques intermédiaires du niveau précédent. Enfin, cette combinaison est utilisée par un générateur de flot $\mathcal{F}_f$, qui est un réseau CNN classique. Ses sorties sont utilisées par un prédicteur $\mathcal{P}$, puis le flot est agrandi avec une couche de convolution-transposition $\mathcal{U}_f$. En parallèle, une couche similaire $\mathcal{U}_h$ agrandit les caractéristiques intermédiaires pour le prochain niveau.

MobileFlow exploite la même structure d'estimation du flot que PWC-Net. Certains modules sont toutefois remplacés par des versions équivalentes optimisées. Toutes les couches de convolution classiques du générateur de flot $\mathcal{F}_f$ et du prédicteur $\mathcal{P}$ sont remplacées par des couches de convolutions séparables, pour alléger le réseau en échange d'une très légère perte de performance sur l'estimation du flot optique. Enfin, les modules $\mathcal{U}_f$ et $\mathcal{U}_h$, chargés d'agrandir le flot et les caractéristiques intermédiaires, passent d'une convolution transposée à des convolutions séparables suivies d'un rééchantillonage bi-linéaire. Ces modifications réduisent le nombre de paramètres entraînables tout en limitant la présence possible d'artéfacts [10].

3 Déploiement vers de l'embarqué

Les réseaux de neurones profonds entraînés dans un environnement sont capables de migrer vers des cibles matérielles par le biais d'un agent intermédiaire. Plusieurs *frameworks* et formats assurent une portabilité inter-plateforme pour chaque phase de déploiement.

MobileFlow est conçu et entraîné sur le framework haut-niveau Pytorch. Une fois entraîné, ses poids et sa topologie sont convertis en format de représentation unifiée ONNX, qui

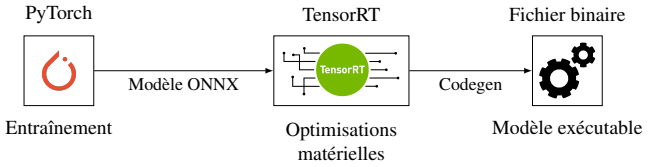

FIGURE 3 – Phases de déploiement de MobileFlow sur la Jetson AGX Xavier.

propose un ensemble de définitions et de types standards pour décrire les opérateurs de réseaux de neurones. Enfin, la description en ONNX est analysée par le framework TensorRT de chez Nvidia. TensorRT applique au modèle des optimisations spécifiques à la cible et recherche une stratégie optimale d'utilisation des ressources matérielles disponibles. Le cheminement global est illustré dans Figure 3.

Cependant, ONNX et TensorRT ne supportent pas nativement les opérations de *backwarp* et de corrélation présentes dans MobileFlow. Nous avons donc codé ces deux couches selon les spécifications ONNX et en language CUDA, puis les avons intégrées en passant par l'API d'ajout de plugins de TensorRT. En particulier, une nouvelle implémentation de la couche de corrélation a été conçue pour respecter le format de stockage des couches de TensorRT (N,C,H,W) qui diffère de celui du framework caffe d'origine (N,H,W,C) de [11]. Le code CUDA a été également amélioré en ayant recours à la mémoire partagée au sein d'un bloc, alors possible avec le format (N,C,H,W). Désormais, le déploiement de PWC-Net et de MobileFlow sur une cible de chez Nvidia est possible.

4 Expériences

Après avoir vérifié leur efficacité, les couches de corrélation $\mathcal{C}$ et *backwarp* $\mathcal{W}$ construites serviront au déploiement de notre implémentation de PWC-Net, comme modèle de référence pour mieux quantifier les gains apportés par l'architecture de MobileFlow.

4.1 Déploiement de référence avec PWC-Net

Une première analyse est menée sur l'efficacité de la couche de corrélation $\mathcal{C}$ créé et optimisée manuellement en CUDA pour TensorRT. Pour cela, PWC-Net est déployé sur une carte GPU Jetson AGX selon le cheminement précédent. Les images d'entraînements et de test proviennent du dataset FlyingChairs. La ligne de commande `trtexec` de TensorRT convertit PWC-Net en un objet *engine* et effectue des optimisations semi-automatiques. Les logs de runtime de TensorRT indiquent que la conversion du coeur CUDA de corrélation du format (N,H,W,C) au format (N,C,H,W) aboutit à un temps d'inférence presque identique (de 19.7 vers 18.7 ms). Cependant, l'utilisation du format (N,C,H,W) combiné à l'utilisation de la mémoire partagée, permet une réduction d'environ 23% du temps d'inférence (15.2 ms).

L'inférence du réseau PWC-Net d'origine fournit déjà des résultats acceptables pour de l'embarqué. L'Endpoint Error (EPE) atteint une valeur de 2.28 pixels. Les performances en temps d'exécution varient de 20.7 FPS (`fp32`) à 36.3 FPS (`fp16`) selon le format de données paramétré avec TensorRT.

4.2 Déploiement du modèle et inférence

Plusieurs versions de MobileFlow sont générées en faisant varier la valeur du paramètre α de profondeur de MobileNetV2, allant de 0.1 (version compacte) à 1 (profondeur inchangée). PWC-Net et ces différents Mobileflow sont entraînés sur Pytorch selon la méthode de *long scheduling* décrite pour PWC-Net [3], avec un taux d'apprentissage dégressif initial de 10^{-4}. Une version pré-entrainée de MobileNetV2 pour l'extracteur est employée directement. Lors de la phase finale avec TensorRT, les réseaux sont optimisés en précision `fp32` et `fp16`, pour les couches le supportant. L'EndPoint Error finale reste la même quelle que soit la précision chosie.

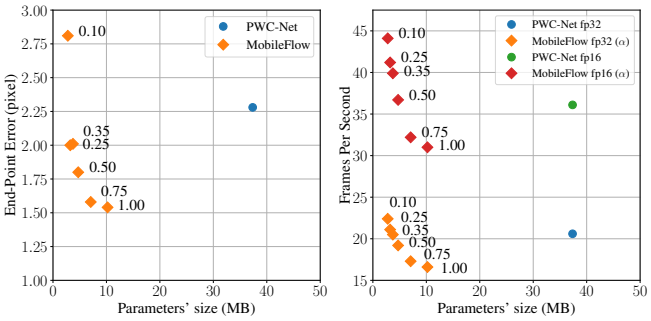

FIGURE 4 – EPE (gauche) et FPS (droite) obtenus sur la Jetson AGX Xavier selon le poids des paramètres.

Par rapport au poids des paramètres, tous les différents MobileFlow générés sont plus légers que le PWC-Net que nous avons entraîné. La bibliothèque Python `torchinfo` estime que le MobileFlow le plus lourd atteint 10.2 MB de paramètres contre plus de 35 MB pour PWC-Net. Les MobileFlow de largeur α supérieure à 0.25 parviennent à une Endpoint Error plus faible que PWC-Net, de 2.0 (-12%) à 1.54 pixels (-32%). Néanmoins, quelle que soit la précision, seuls les MobileFlow les plus légers (α inférieur à 0.35) ont un nombre d'images par seconde (FPS) plus important que PWC-Net, d'au plus 22.4 contre 20.6 FPS (`fp32`) et 44.1 contre 36.1 FPS (`fp16`). Ainsi, les performances des MobileFlow ne dominent pas systématiquement celles de PWC-Net. Néanmoins, les MobileFlow forment un ensemble de compromis, comme illustré par les fronts de Pareto en Figure 4. MobileFlow-0.25 est une des versions dominant PWC-Net à la fois sur l'EPE (2.0, -12%), le poids des paramètres (3.23 MB, -91%) et le temps d'inférence final en précision `fp16` (41.2 FPS, +14%).

5 Analyse

Les vitesses d'estimation du flot optique (FPS) obtenues avec les MobilesFlow sont moins bonnes qu'espérées initialement, surtout pour les versions les plus compactes ($\alpha < 0.5$), censées permettre un allègement conséquent du réseau. Une analyse complémentaire est menée pour identifier quels facteurs limitent la réduction du temps d'inférence.

5.1 Analyse du réseau

Au sein du réseau, une mauvaise répartition du nombre de MAC (Multiply-ACcumulate) et du nombre de paramètres par couche peut être source de mauvais équilibrage des charges dans le pipeline logiciel (*workload*) lors de l'inférence. La librairie `torchinfo` propose une estimation de ces paramètres pour les couches de réseaux de neurones usuelles. Seul le nombre de MACs des couches de corrélation $\mathcal{C}$ et de *backwarp* $\mathcal{W}$ a été estimé manuellement, leur nombre de paramètres étant nul ici. Cette analyse mène à deux constats :

— Pour chaque MobileFlow, les couches de convolution de l'estimateur de flot optique demandent le plus de paramètres et d'opérations,

— En particulier, deux couches de convolution à la fin de l'estimateur de flot optique représentent de 28% (α = 0.10) à 50% (α = 1) du nombre total de MACs. Une explication possible viendrait de l'augmentation de la profondeur des couches à travers le réseau, créant des convolutions très profondes sur les dernières couches. La profondeur de ces couches est pourtant nécessaire. La diminuer dégrade fortement l'estimation du flot optique.

5.2 Profiling sur l'inférence

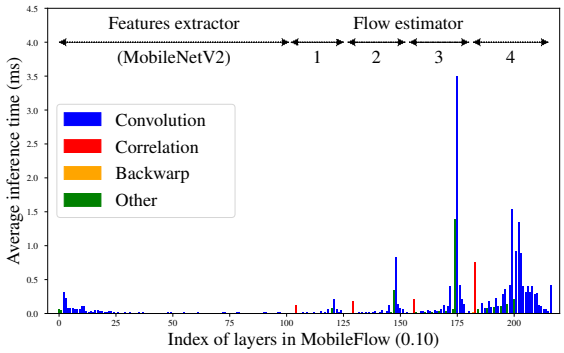

FIGURE 5 – Inférence par couche sur la Jetson AGX Xavier pour MobileFlow α = 0.10 en `fp32`

En fonction de la cible matérielle, la relation entre le nombre de MACs et de paramètres et le temps d'inférence effectif n'est pas forcément directe. Il convient d'analyser aussi le temps d'inférence par couche grâce aux fichiers logs fournis par TensorRT ou par Nsight-Compute (Nvidia) sur la Jetson Xavier. L'analyse devient plus complexe, car chaque configuration stagne sur une couche particulière, mais il semble néanmoins que :

— Les couches en fin de l'estimateur de flot optique présentent les latences les plus élevées,

— Comparé à la précision `fp32`, celle en `fp16` diminue essentiellement la latence des couches de convolution,

— Le paramètre de profondeur α diminue la latence globale de toutes les couches.

Les niveaux 3 et 4 du sous-réseau estimateur de flot sont des obstacles possibles à la réduction de la latence.

6 Conclusion

Dans cet article, nous proposons MobileFlow, un réseau CNN compact d'estimation de flot optique pour des applications embarquées, basé sur le réseau PWC-Net. Le réseau MobileFlow est plus léger que le PWC-Net d'origine grâce à différents remplacements opérés. Une fois entraîné, Mobileflow suit un cheminement à travers plusieurs phases de déploiement pour être testé sur une carte GPU Jetson AGX. L'implémentation de couches spécifiques supplémentaires a été nécessaire et assure une portabilité entre chaque phase. Les différentes versions de MobileFlow optimisés obtenues offrent un compromis entre la performance d'estimation de flot optique et les contraintes de l'embarqué (temps-réel, poids des paramètres). En particulier, sur le dataset Flying Chairs, MobileNet-0.25 est plus performant de 12% sur l'estimation du flot optique, plus léger de 91%, plus rapide de 14% en précision `fp16` que PWC-Net.
Un profiling complémentaire a déterminé que l'inférence rencontre une mauvaise répartition des *workload* dans le sous-réseau estimateur de flot optique de MobileNetV2. Les temps d'inférence de MobileFlow pourraient être améliorés en allégeant encore principalement son deuxième sous-réseau, ou avec une quantification INT8.

Références

[1] B. K. P. HORN et B. G. SCHUNCK. "Determining optical flow". en. In : *Artificial Intelligence* 1 (août 1981).

[2] A. DOSOVITSKIY et al. "FlowNet : Learning Optical Flow with Convolutional Networks". In : *2015 IEEE/ICCV*.

[3] D. SUN et al. "PWC-Net : CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume". In : *2018 IEEE/CVF*. Juin 2018, p. 8934-8943.

[4] A. G. HOWARD et al. "MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications". In : *arXiv :1704.04861 [cs]* (avr. 2017).

[5] M. SANDLER et al. "MobileNetV2 : Inverted Residuals and Linear Bottlenecks". In : *arXiv :1801.04381 [cs]* (mars 2019).

[6] A. HOWARD et al. "Searching for MobileNetV3". en. In : *arXiv :1905.02244 [cs]* (nov. 2019).

[7] F. N. IANDOLA et al. "SqueezeNet : AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". In : *arXiv :1602.07360 [cs]* (nov. 2016).

[8] S. HAN et al. "Learning both Weights and Connections for Efficient Neural Networks". In : *arXiv :1506.02626 [cs]* (oct. 2015).

[9] D. D. LIN et al. "Fixed Point Quantization of Deep Convolutional Networks". In : *arXiv :1511.06393 [cs]* (2016).

[10] A. ODENA et al. "Deconvolution and Checkerboard Artifacts". In : *Distill* (2016).

[11] E. ILG et al. "Caffe for FlowNet2". In : *https ://github.com/lmb-freiburg/flownet2* (2019).