Rapport
Projet applicatif Vidéo

Jenny Benois-Pineau, Boris Mansencal

Jules Civel, Nathan Trouvain
ENSC 3A IA

2019 - 2020

Sommaire

Introduction

Le traitement de données issus de flux vidéos est un enjeu de nombreuses applications pratiques des techniques d'intelligence artificielle. Nous nous intéressons ici en particulier aux systèmes d'assistance, de suppléance et d'augmentation de l'humain, dont une des problématiques associées est la compréhension de l'intention de l'opérateur humain, en contexte et en simultané de l'action, pour permettre de répondre à sa commande et à son besoin de manière adéquate.

Dans le cadre de l'utilisation de prothèses de bras, choisi comme contexte pour ce projet, l'analyse d'informations visuelles issues du point de vue de l'utilisateur pourrait par exemple permettre de rendre l'action du bras artificiel conforme à l'intention de l'utilisateur. En interprétant les données vidéos reçues d'un capteur placé au niveau du bras synthétique, on cherche alors à interpréter l'action en cours de l'utilisateur, en particulier son attitude vis à vis des objets présents dans son champ de vision. L'utilisateur pourrait ainsi contrôler la prothèse pour la préhension d'objet simplement en orientant son champ de vision vers l'objet, de la même manière que le ferait une personne "valide" dans les secondes précédant la préhension d'un objet.

Le traitement et l'interprétation de ces flux vidéos impliquent l'utilisation de techniques de vision par ordinateur efficaces, capables de produire des représentations pertinentes de l'environnement de l'utilisateur. Ces représentations peuvent être obtenues à l'aide de techniques de *deep learning*, notamment l'utilisation de réseaux convolutifs profonds.

Cette tâche étant de plus très liée au contexte d'utilisation de la potentielle prothèse, il est nécessaire de disposer d'échantillons de données représentatifs de l'environnement que l'on souhaite interpréter. A cet effet, nous disposons pour cette tâche d'un jeu de données spécifique, *Grasping In The Wild* (GITW).

En combinant des techniques de *deep learning* obtenant des résultats satisfaisants dans la littérature et le jeu de donnée mis à disposition, nous présentons dans ce rapport le

processus de construction d'un modèle permettant d'interpréter l'objectif d'un potentiel utilisateur de prothèse à l'aide d'informations visuelles.

Présentation du problème

Reconnaissance d'images

La reconnaissance d'activité que l'on souhaite mettre en place au regard du contexte de ce projet se base sur une compréhension la plus fine possible de l'environnement de l'utilisateur. Une des composante majeure de cette compréhension est la reconnaissance des objets présents dans cet environnement. Cette reconnaissance doit avoir lieu quelque soit le contexte de l'objet: sa position, sa taille relative, ses possibles rotations, et des contraintes extérieures à l'objet telles que la luminosité, les objets avoisinant, ou le masquage par d'autres objets. Le modèle conçu doit en effet *in fine* pouvoir reconnaître l'objet visé par l'utilisateur dans l'image, quelque que soit le contexte.

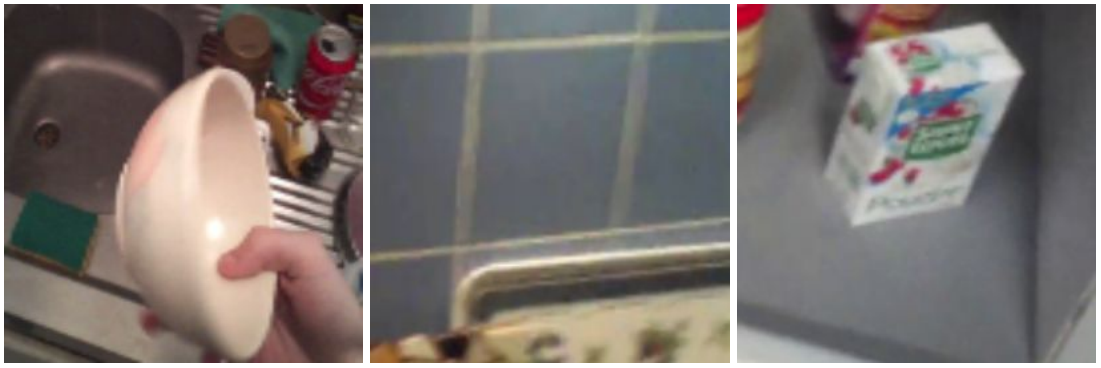
On souhaite donc, à partir d'un jeu de données contenant des images de différents objet, reconnaître ces objets quelque soit leur position dans l'espace ou leur orientation.

Les réseaux de neurones convolutifs sont connus dans la littérature pour être particulièrement pertinents dans cette situation. Ils permettent d'extraire d'un corpus d'images des représentations locales qui ne sont pas affectées par les effets d'échelle ou par les translations dans l'espace. Ils permettent également, lorsque combinés au sein de réseaux profonds, d'extraire des informations perceptuelles relativement abstraites, qui peuvent ensuite être utilisées pour représenter l'image d'entrée de manière pertinente. Ces représentations de haut niveau d'abstraction peuvent être traitées en particulier pour la reconnaissance des objets présents dans l'image, au sein d'une tâche d'apprentissage supervisé. Nous nous intéresserons donc à ce type de réseaux de neurones dans la suite de ce rapport.

Jeu de données *GITW*

L'apprentissage supervisé de la tâche de reconnaissance d'image que l'on se propose d'effectuer est effectué sur le jeu de données *Grasping In The Wild* (GITW) (*Figure 1)*. GITW est composé d'images issus de vidéos enregistrées depuis un point un de vue subjectif dans une cuisine encombrée d'objets du quotidien. On peut y voir 16 catégories d'objets différents être regardés puis saisis par un sujet humain se déplaçant dans la cuisine. Une dernière catégorie regroupe des images ne contenant pas d'objet d'intérêt. Ces images représentent un échantillonnage des vidéos originales à hauteur de 1 image sur 40.

Nous utiliserons dans le cadre de ce projet le jeu de donnée *GITW_clean*, produit par un autre groupe d'étudiants, dont les images ont été recadrées autour des objets d'intérêt. Ce jeu de données permettra de sensiblement aider l'apprentissage en le concentrant sur les zones d'intérêts des images d'entrées.



a. b. c.

Figure 1. Exemples de données contenues dans *GITW_clean* (catégories: a. *Bowl,* b. *Background*, c. *Sugar*)

Réseaux convolutifs pour la classification d'images

Transfer Learning

L'entraînement de réseaux convolutifs capables de produire des représentations latentes suffisamment générales pour convenir à des problèmes de classification généraux est trop coûteux pour être effectué dans le cadre de ce projet. Nous préférerons donc produire un modèle par *transfer learning*.

Le *transfer learning* est une méthode d'apprentissage machine permettant de "sur-entraîner" un modèle général dans le but d'effectuer une tâche plus spécifique. Les représentations latentes apprises sur des jeux de données très importants de modèles pré-conçus sont ainsi sauvegardées, évitant d'effectuer un nouvel entraînement complet, qui nécessite une grande puissance de calcul et un jeu de données très important.

L'entraînement consiste alors à ajuster un certains nombres des paramètres d'un modèle déjà entraîné sur une tâche de classification d'image générale, et à utiliser les représentation produites par ces ajustements au sein d'un réseaux de neurones, entraîné spécifiquement pour la classification qui nous intéresse.

Nous avons choisi pour ce projet *ResNet* et *VGG19* comme bases pour le *transfer learning*, deux réseaux de neurones entraînés sur *ImageNet*, un corpus de plus de 100 000 images, sur une tâche de classification autour de 1000 catégories d'objets.

Ces deux réseaux traitent des images de dimensions 224x224 pixels, codées au format RGB.

ResNet

ResNet50 (Residual Network) (*Figure 2*) est un réseau de neurone convolutif obtenant une précision de 92% (top 5) sur le corpus *ImageNet*. Il compte 25 millions de paramètres.
ResNet est conçu de telle manière à permettre à de l'information résiduelle brute de descendre en profondeur dans les couches convolutives du réseaux. Cette information permet de maintenir une valeur du gradient significative tout au long de la transformation des informations en entrée.

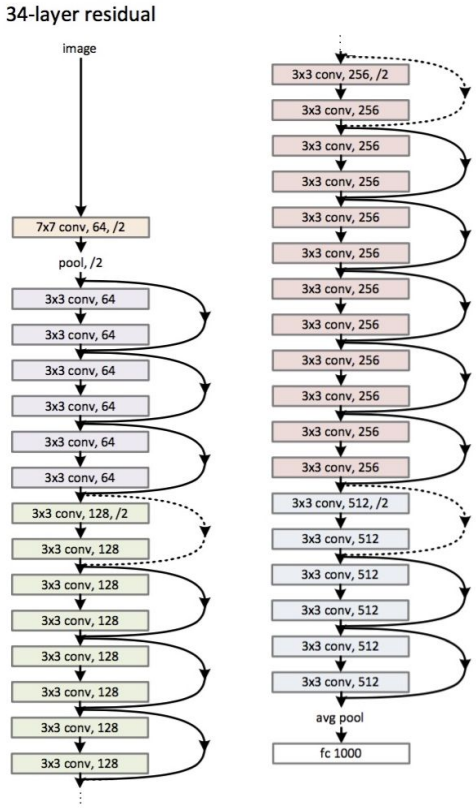


Figure 2. Architecture ResNet50. L'information est réinjectée de chaque bloc de convolution *n* vers le bloc *n+1.*

VGG19

VGG19 (Visual Geometry Group) (*Figure 3*) est un réseau de neurone convolutif obtenant une précision de 90% (top 5) sur le même corpus *ImageNet*. Il compte 144 millions de paramètres. VGG19 base son architecture sur l'exploitation de blocs de convolution. Chaque bloc extrait des informations de plus en plus abstraites en fonction de sa profondeur dans le réseau. ResNet peut être vu comme une amélioration de VGG apportant le concept d'informations résiduelles. VGG19 est cependant plus simple à manipuler, l'information ne descendant dans le réseau que par une seule voie directe.

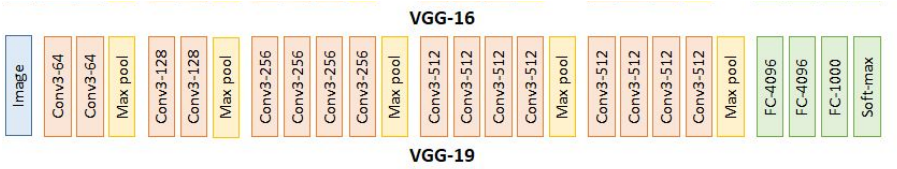


Figure 3. Architecture VGG19. Le réseaux est structuré en 5 blocs de convolutions, séparés par des opérations de seuillage (*max pooling*).

Entraînement

Au dessus de ces deux réseaux est ajouté un perceptron multicouche permettant la classification suivant les 17 catégories de GITW. L'apprentissage est effectué par optimisation d'une fonction d'entropie croisée à l'aide de techniques de descente stochastique de gradient. Les paramètres sont mis à jour suivant l'algorithme Adam, qui exploite le gradient ainsi que les premiers moments pour la minimisation de la fonction de perte. Le coefficient d'apprentissage associé est diminué à chaque itération d'un facteur 1.10^{-8}, et est réduit d'un facteur 1.10^{-1} lorsque 10 itérations voient la perte liée aux données de test stagner.

L'entraînement utilise les bibliothèques d'apprentissage machine *Keras* et *TensorFlow*. Le suivi des entraînement est stocké dans le dossier *logs* du code source, et est au format *TensorBoard.*

Les métriques sont calculées sur une partition du jeu de donnée de test non utilisée pour l'entraînement. Cette partition représente environ 15% du jeu de données original.

Résultats

Transfer Learning sur ResNet50

L'entraînement est effectué sur 20 itérations d'apprentissage, avec l'algorithme d'optimisation Adam. Les précisions obtenues au cours de l'entraînement sont récapitulées en *Figure 4.*

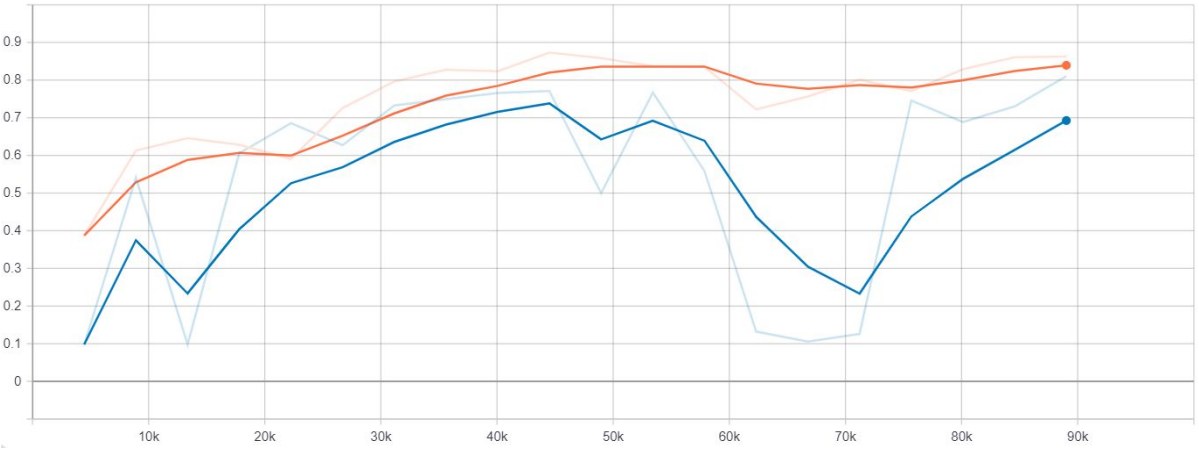


Figure 4. Précision d'entraînement (orange) et de test (bleu) obtenues lors du *transfer learning* sur ResNet.

La précision maximale de test est 68% sur GITW.

On remarque d'importantes fluctuations au cours de l'apprentissage. Celles-ci peuvent s'expliquer par la stratégie d'adaptation du coefficient d'apprentissage choisie. Le facteur de diminution du coefficient et le déclenchement de la stratégie de diminution sont deux hyperparamètres qu'il serait nécessaire d'explorer plus avant. De bien meilleurs résultats pourraient ainsi être obtenus.

En raison de limitations techniques liés à la disponibilité de machines suffisamment puissantes pour effectuer les calculs, seule un apprentissage a pu être mené à terme.

Transfer Learning sur VGG19

Le surentraînement de VGG19 a été conçu en limitant l'ajustement des paramètres aux dernières couches convolutives du réseau. Ainsi, les représentations latentes les plus locales - et donc les plus générales, car concernant la texture de l'image plus que son contenu - sont conservées lors de l'entraînement. Seules les représentations latentes de plus haut niveau sont réentraînées. Le coût de l'entraînement est ainsi diminué, et celui-ci est concentré sur les couches d'intérêt.

Entraînement sur les blocs 4 et 5

Un premier entraînement a été effectué en ne sur-entraînant sur GITW que les deux derniers blocs convolutifs de VGG19, les désignés dans le modèle comme blocs 4 et 5. Tous les autres paramètres du modèles, hormis le perceptron de sortie, sont maintenus constants au cours de l'apprentissage. Les précisions obtenues au cours de l'entraînement sont récapitulées en *Figure 5.*

Cet entraînement permet d'atteindre au bout de 20 itérations une précision de test maximale de 91%.

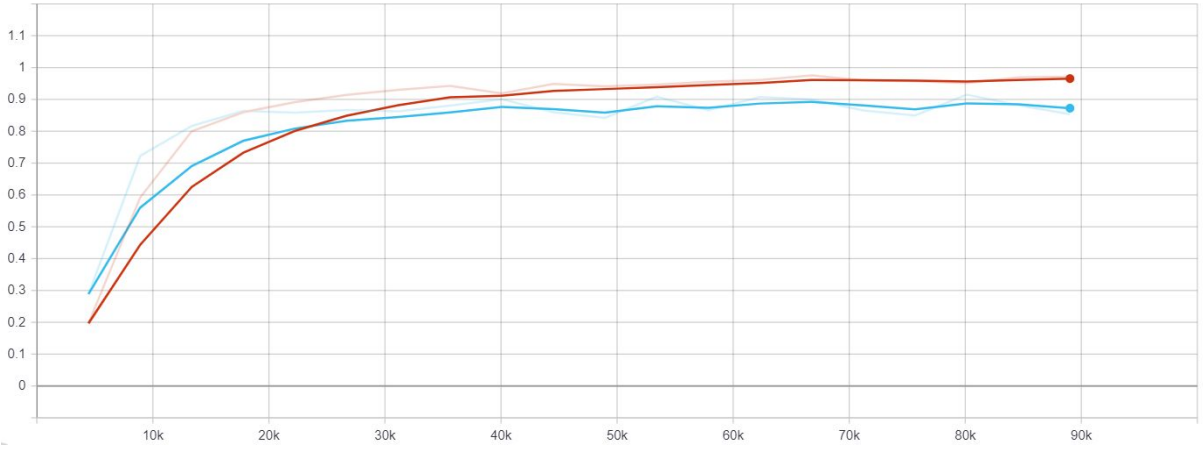


Figure 5. Précision d'entraînement (rouge) et de test (bleu) obtenues lors du *transfer learning* sur VGG19 blocs 4 et 5.

Entraînement sur le bloc 5

Un second entraînement est effectué en ne sur-entraînant que le dernier bloc du VGG19 sur GITW. Cet entraînement a permis d'atteindre une précision de test maximale de 92%. Les précisions obtenues au cours de l'entraînement sont récapitulées en *Figure 6.*

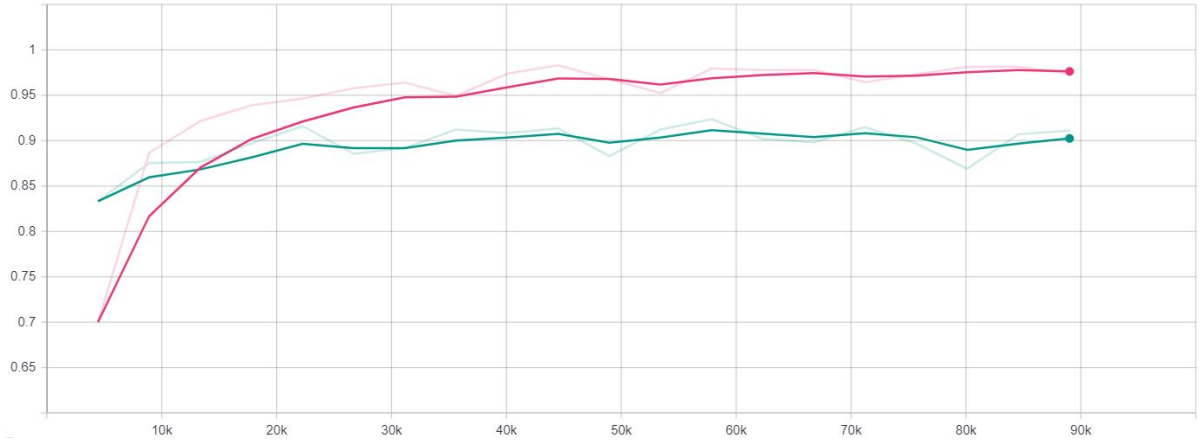


Figure 6. Précision d'entraînement (rose) et de test (vert) obtenues lors du *transfer learning* sur VGG19 bloc 5.

Discussion

L'ensemble des résultats obtenus sont récapitulés dans le *Tableau 1*.

Modèle	Taux d'apprentissage (optimisation *Adam*)	Précision maximale (top 1)
ResNet50	1.10^{-3}	68%
VGG19 bloc 4 et 5	1.10^{-4}	91%
VGG19 bloc 5	1.10^{-4}	92%

Tableau 1. Récapitulatifs des résultats obtenus avec les architectures ResNet, VGG19 bloc 4 et 5 et VGG19 bloc 5.

Peu de résultats ont pu être produits, en raison de limitations importantes sur la puissance et le temps de calcul. Les architectures développées et les entraînements effectués doivent

donc être vues comme des preuves de concept, et nécessiteraient une exploration bien plus importante des hyperparamètres pour ajuster les modèles.
De plus, en raison de ces mêmes limitations techniques l'entraînement a été effectué à l'aide de paramètres défavorables à la qualité du projet. La taille des *batch* en entrée des réseaux à part exemple été fixée à 8 lors de l'entraînement des VGG, ce qui est potentiellement vecteur d'un important biais sur le calcul des gradients d'erreur lors de l'optimisation de la fonction de perte. Le changement de type d'architecture - de ResNet vers VGG - est également facteur de confusion dans les résultats, et ne s'explique que par le manque de puissance de calcul dans la deuxième partie du projet, qui a été menée sur nos machines personnelles.

Il serait en particulier pertinent d'explorer le nombre minimal de couches à réentraîner dans les réseaux VGG et ResNet lors du transfert. La modification de ces paramètres, en plus de permettre d'économiser un nombre important de ressource, pourrait également ouvrir la voie à des entraînement *on-line* rapides, au sein desquels seuls une petite partie du réseau est réentraînée à chaque étape, sur des échantillons de données restreints. Si les résultats sur VGG semblent indiquer qu'une telle réduction des paramètres d'entraînement est possible, elle est à confirmer et à tester sur ResNet.

On constate néanmoins que ces architectures permettent d'obtenir des résultats satisfaisants. De plus, toujours dans la perspective de l'analyse d'un flux vidéo et donc d'un apprentissage *on-line*, il apparaît que le temps de réentrainement pour jusqu'à obtention d'une précision correcte est relativement faible. VGG19 bloc 5 est par exemple à 80% de précision dès les premières étapes d'entraînement, comme visible sur la *Figure 7.*

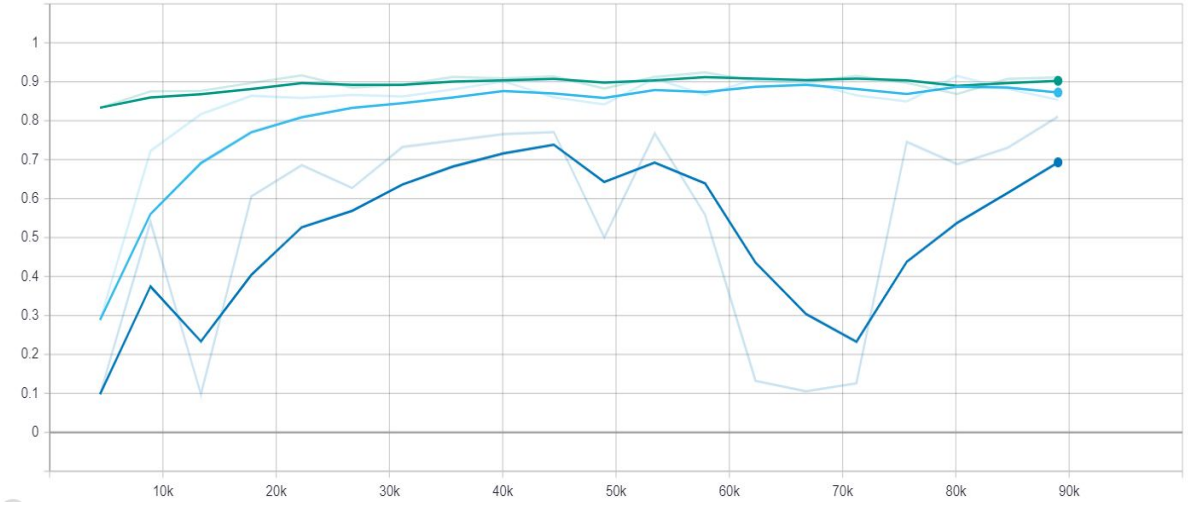


Figure 7. Précisions de validation comparées de ResNet (bleu foncé), VGG19 blocs 4 et 5 (bleu clair) et VGG19 bloc 5 (vert).