Université Laval

GIF-7001

Vision numérique

Projet de trimestre : Contrôle par le geste (de la main ou du bras)

Équipe :	*Numéros d'identification :*
Olivier Gamache	536787687
Damien LaRocque	111279741
Gabriel St-Pierre	111178727

Remis à
Pr Denis Laurendeau, ing., Ph.D

Le 14 décembre 2020

1 Introduction

Le but de ce projet est de concevoir un lecteur vidéo qui réagit à des gestes de la main effectués devant une webcam. Pour ce faire, il fut divisé en 3 sections principales. Premièrement, il faut prendre une image avec la webcam de l'ordinateur et y relever la région contenant la main. Deuxièmement, il faut détecter les gestes de la main. Enfin, une fois identifié le signe de la main, il ne reste qu'à y relier une action ayant une influence sur la vidéo, comme le fait de la pauser ou de la mettre en sourdine, par exemple. Ces différentes parties de l'algorithme du projet sont montrées à la figure 1. Il est à noter que, pour faciliter son développement, le projet a été programmé en Python avec l'aide, notamment, des programmathèques `OpenCV` et `PyTorch`.

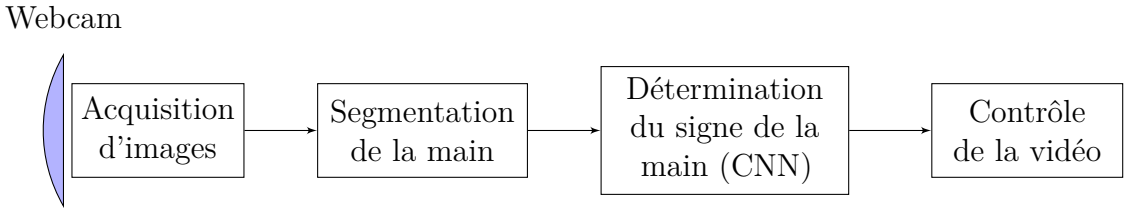

Figure 1 – Flux opérationnel de l'algorithme

2 Description de la solution proposée

Les sous-sections suivantes décrivent en détail les différentes composantes de l'algorithme de projet.

2.1 Segmentation de la main

Pour identifier la main dans l'image et pour l'isoler du restant de l'image, de la segmentation fut réalisée à l'aide de la bibliothèque `OpenCV`. Tout d'abord, l'image fut convertie dans l'espace colorimétrique TSV (*HSV*, en anglais), représenté par trois composantes : la teinte, la saturation et la valeur. Chacune de ces composantes fut bornée avec des valeurs permettant de caractériser des objets et de les relever dans la scène. Ainsi, dans le cas présent, les composantes furent limitées pour récupérer les pixels ayant une couleur se rapprochant de celle de la peau de la main. L'algorithme sélectionne aussi la plus grande section dans l'image (qu'on suppose étant la main). Ensuite, un masque binaire fut réalisé à partir de ce filtrage ce qui a permis de délimiter une zone autour de la main, comme le montre la figure 2(B). Par la suite, la fonction `cv2.findContours()` a permis de déterminer les arêtes d'illuminance sur le masque binaire, permettant ainsi de délimiter le contour de la main. Ce contour a permis de délimiter sur l'image un rectangle contenant la main, comme présenté à la figure 2(C). Enfin, ce rectangle est isolé de l'image d'origine, comme affiché à la figure 2. C'est cette dernière image qui sera envoyée dans le réseau, après quelques transformations en échelle et en couleur. Le processus complet est illustré à la figure 2.

2.2 Détection des signes dactylologiques (Langue des signes)

Afin de détecter les signes dactylologiques de la main, le réseau de neurones à convolution fut choisi pour son efficactié et pour son adaptation aux variations de couleur, de luminosité et de fond.

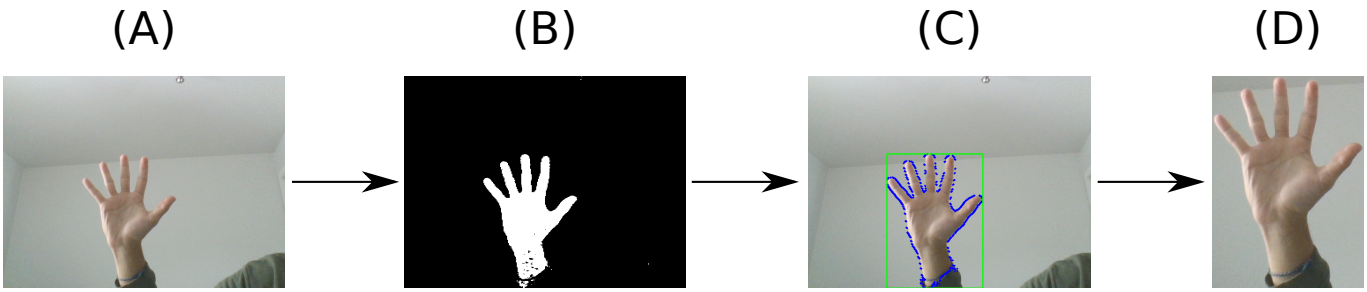

Figure 2 – Illustration des étapes de segmentation de la main, de l'image originale (A), après filtrage de couleurs (B), après la détection de contours (C) et avant envoi au réseau de neurones (D).

La mise en place d'un tel réseau se découpe en trois étapes principales, qui seront décrites dans les sous-sections suivantes.

2.2.1 Données

Au niveau des données, il fut décidé d'utiliser un jeu de données existant, puisque ces derniers offrent une bonne variété d'environnements différents. Celui qui a été choisi porte sur la langue des signes américaine, qui comprend une bonne gamme de signes facilement distinguables. On y retrouve 24 classes correspondant à 24 lettres de l'alphabet (excluant J et Z qui nécessitent des mouvements). Il comporte 27455 images d'entraînement et 7172 images de test de taille 28×28 pixels et en niveaux de gris [1].

Afin de permettre une meilleure généralisation des données et pour éviter que le réseau ne se fie que sur les détails fins des doigts, une série de manipulations préliminaires est effectuée lors de l'alimentation des données :

- Réflexion horizontale
- Effacement aléatoire (10 % à 33 %)
- Rotation aléatoire (±15 degrés)
- Translation aléatoire (max 10 %)
- Zoom aléatoire (90 % à 110 %)

Finalement, puisque plusieurs signes se ressemblent, ils ont été catégorisés dans des super-classes :

1. Poing : A, E, M, N, O, S et X
2. Ouvert : B et C
3. Pointe vers le haut : D, K, I, R, T et U
4. Pointe vers le bas : P et Q
5. Pointe sur le côté : G et H
6. Signe en V : V et W
7. Classes uniques : Y, L, F (main écartée)

2.2.2 Réseau

Puisque les images sont de nature plutôt simple, il a été décidé de construire un réseau maison afin d'effectuer l'entraînement. L'architecture de ce dernier utilise des méthodes standards et modernes afin de classifier les signes, comme illustré à la figure 3.

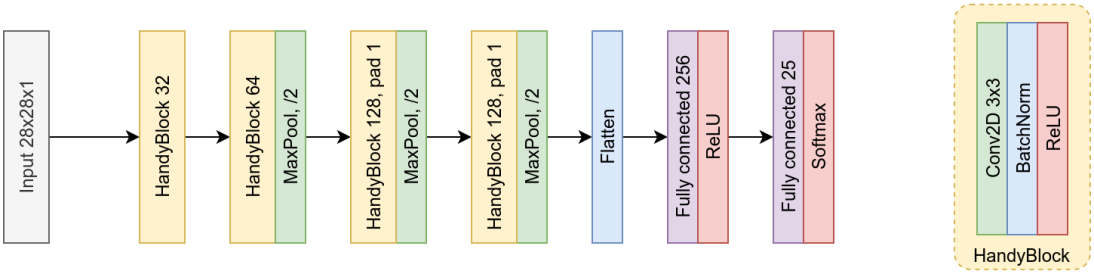

Figure 3 – Architecture du réseau HandyNet

Afin de trouver le signe correspondant, il suffit de trouver l'indice de la valeur maximale de la sortie du réseau. Afin d'éviter une association absolue, la sortie a été seuillée, de telle sorte que le réseau ne retourne rien s'il n'y a pas une confiance minimale sur la sortie.

2.2.3 Entraînement

L'entraînement s'effectue sur un maximum de 30 *epochs* (la plus performante en test étant celle qui sera sauvegardée), sur 90 % du jeu d'entraînement et à l'aide de l'optimiseur *Adam*. La mesure d'erreur *CrossEntropy* a été utilisée afin d'entraîner le réseau rapidement et efficacement. Finalement, une diminution automatique du taux d'apprentissage permet au réseau de s'ajuster en cas de divergence. Le taux d'apprentissage initial est de 0.001. Chaque *epoch* s'effectue sur des *batchs* de 128 images afin d'aider à la généralisation sans toutefois trop affecter le temps d'exécution.

2.3 Contrôle de la vidéo

Cette section du projet fut réalisée en deux phases. La première de ces phases consistait à réaliser un code pour lire une vidéo, alors que la seconde phase consistait à utiliser le résultat de la classification avec le réseau de neurones pour contrôler la vidéo.

Le logiciel `VLC Media Player` [2] fut choisi pour lire la vidéo. Ce logiciel a comme avantage d'être gratuit et libre, permettant par le fait même son installation sur un système d'exploitation reposant un noyau Linux. De plus, puisqu'il est libre, il était plus probable qu'il existe une programmathèque en Python permettant de contrôler un lecteur vidéo, comme la bilbiothèque `python-vlc` [3] qui contient une interface de programmation [4] pour contrôler une instance de VLC. Cette interface de programmation a permis de créer une classe permettant d'instancier des objets imitant le fonctionnement d'un lecteur vidéo. Il fut ensuite possible de créer une autre classe définissant un contrôleur de lecteur vidéo dont le rôle était de contenir ce lecteur et de définir les commandes à envoyer à celui-ci à partir du signe détecté par le réseau de neurones.

Pour éviter d'envoyer trop de commandes à la fois au lecteur vidéo, une temporisation de trois secondes fut ajoutée au contrôle. Ainsi, seule une instruction est envoyée à chaque trois secondes, à partir des signes de la main accumulés durant cette période. La sélection de cette instruction se base sur un scrutin majoritaire uninominal : est uniquement considéré le signe de la main qui fut majoritairement détecté durant le temps de temporisation.

Une fois la commande déterminée, elle est mise en correspondance avec une méthode du lecteur vidéo implémenté plus tôt.

3 Performances

Une vidéo illustrant le résultat final de ce projet est disponible sur Youtube au lien suivant : `https://youtu.be/54B7buLo7X0`

Comme pour la majorité des applications en vision par ordinateur, les performances de notre projet dépendent fortement de l'illuminance et de la caméra utilisée. Cette illuminance dépend elle-même de la luminosité et du contenu de la scène. Ainsi, il fut remarqué que les performances de l'application étaient nettement supérieures lorsque la main était bien éclairée et lorsque l'arrière-plan est de couleur unie (comme un mur blanc, par exemple). Malgré tout, la segmentation de la main fonctionne la majorité du temps, si aucune autre partie de peau (visage, bras, par exemple) n'est affichée face à la caméra.

Concernant les performances du réseau HandyNet, une précision de 93 % a été atteinte sur le jeu de test. La figure 4 à l'annexe A montre la matrice de confusion résultante. Malheureusement, un tel classement est peu fiable, puisque les résultats dans le monde réel ont plus de variance que les exemples des jeux d'entrainement et de test. Notamment, le réseau a des difficultés à généraliser face à des variations d'éclairage de la main : une main sombre ou à contre-jour performe très mal comparé à une main claire bien éclairée.

Le contrôle de la vidéo, quant à lui, fonctionne plutôt bien. Lorsqu'un geste de la main est reconnu parmi les gestes acceptés, une commande est envoyée à la vidéo. Il subsiste quand même un problème au niveau de la sélection de la commande à envoyer. En effet, pour faire cette sélection, tous les gestes reconnus durant 3 secondes sont pris en compte de manière égale : tant ceux au début qu'à la fin de cette période. Il pourrait être judicieux d'ajouter un poids à la sélection pour que les derniers gestes effectués durant la temporisation aient plus d'importance que les premiers.

4 Améliorations

Plusieurs améliorations sont possibles au niveau de la segmentation de la main. En effet, la segmentation est surtout fondée sur un filtrage de la couleur de la main. Or, cette dernière est notamment sensible à la lumière de la scène, mais elle est aussi aux variations de la couleur de la peau humaine. L'algorithme n'étant pas capable de généraliser pour différentes couleurs de peau, il serait pertinent de développer une méthode qui ne se base pas sur la couleur, mais directement sur la structure de la main. À ce sujet, une tentative d'utilisation de l'algorithme de Canny fut réalisée, mais elle ne s'est pas relevée concluante.

De plus, l'algorithme de segmentation rogne directement le rectangle entourant la main. Il pourrait être préférable d'ajouter un rebord autour de la main pour avoir une meilleure détection par le réseau de neurones comme une partie des doigts est parfois rognée durant l'acquisition d'images.

Au niveau de l'entraînement du réseau, un entraînement portant sur les groupes de signes plutôt que sur les signes individuels pourrait améliorer la performance ainsi que la stabilité de la détection. De plus, une transformation des données en segmentation binaire (main et fond en 2 couleurs unies différentes) aurait certes permis un apprentissage plus efficace, mais aurait été compliqué à effectuer

(en raison de la grande variété d'illuminance et de contrastes au niveau du jeu).

Enfin, en ce qui concerne le contrôle de flux vidéo, comme expliqué à la section 3, il serait intéressant de penser à d'autres stratégies de désignation de la commande à envoyer au flux vidéo. Cela pourrait notamment inclure une pondération pour accorder plus d'importance aux gestes reconnnus durant les dernières secondes de la temporisation.

5 Conclusion

In fine, il fut possible de concevoir un algorithme permettant de contrôler la lecture d'une vidéo à partir de signes de la main exécutés devant une webcam. Si cet algorithme est performant avec des images venant de jeux de données, il semble avoir des difficultés dans la réalité. Cela pourrait être causé par une différence de qualité entre les images d'entraînement et celles prises par la webcam. De même, il se pourrait que le réseau de neurones entraîné et utilisé dans le projet soit surentraîné, entraînant *de facto* des problèmes de généralisation.

6 Références

[1] tecperson, "Sign Language MNIST," Oct. 2017.

[2] VideoLAN Organization, "VLC media player," July 2020.

[3] Python Package Index, "python-vlc 3.0.11115," July 2020.

[4] O. Aubert, "Python bindings for VLC API Documentation," Oct. 2019.

Annexes

A Annexe - Matrice de confusion

La matrice de confusion d'un réseau de neurones permet d'illustrer la capacité de perception du réseau. Dans cette annexe, on retrouve celle de notre réseau HandyNet à la figure 4.

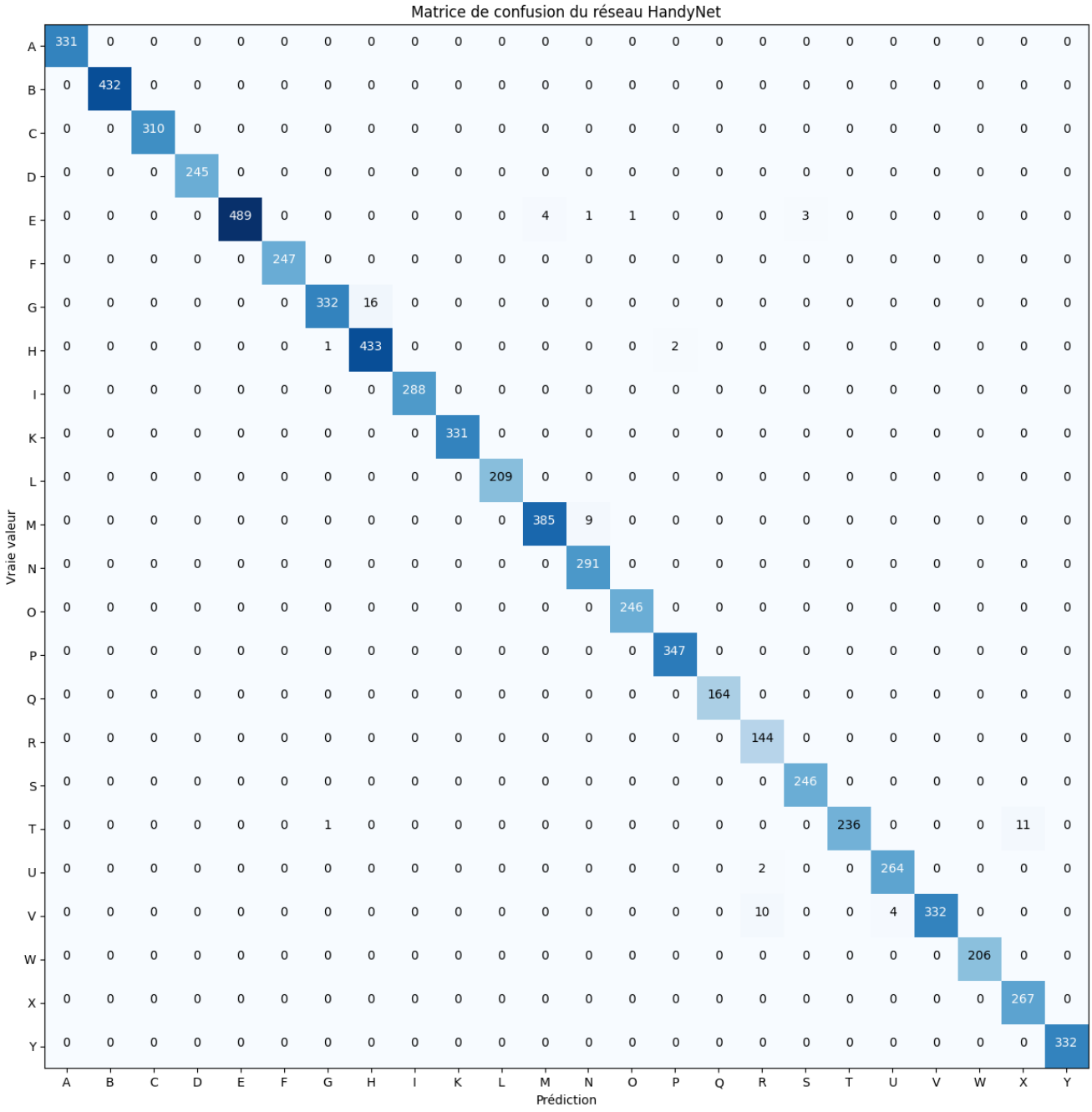

Figure 4 – Matrice de confusion sur le jeu de test du réseau HandyNet