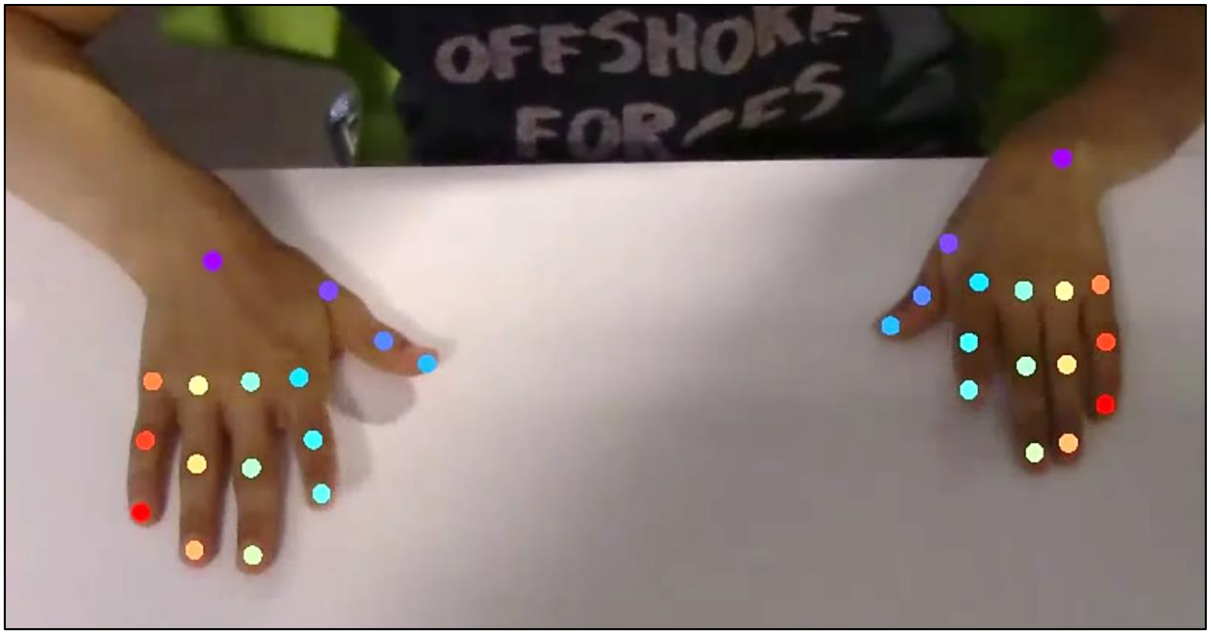
Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Travail de Bachelor 2021

Using DeepLabCut to detect and quantify mirror movements in videos of children with cerebral palsy



Étudiant : Brice Berclaz

Professeur : Dr Henning Müller

Déposé le : 4 août 2021

SOURCE DE L'ILLUSTRATION DE LA PAGE DE TITRE

Photo de l'auteur

RÉSUMÉ

Ce travail de Bachelor a pour but l'expérimentation de la quantification des mouvements en miroir sur des vidéos d'enfants atteints de paralysie cérébrale unilatérale grâce à un logiciel d'estimation de pose nommé DeepLabCut.

Pour commencer, les techniques existantes de diagnostic de mouvements en miroir seront recensées. L'intérêt de ce rapport sera ensuite démontré avec une attention particulière sur la portabilité de l'estimation de pose et ses enjeux.

Ensuite, un modèle de détection et de suivi de mouvements sera entraîné pour apprendre à reconnaître des gestes. L'analyse vidéo d'un exercice défini dans ce rapport permettra de récolter les coordonnées des points clés choisis sous forme de données brutes. Celles-ci seront ensuite transformées pour procéder à une quantification à l'aide de méthodes de corrélations.

Pour terminer, les connaissances acquises dans la mise en place de ce procédé seront rendues accessibles avec une application. Cette interface graphique devra donner l'opportunité aux cliniciens de s'en servir librement pour effectuer leur analyse. À l'aide des résultats, ils pourront confirmer la présence de mouvements en miroir chez l'enfant et consolider leur diagnostic.

Mots-clés : *Deep Learning*, mouvements en miroir, estimation de pose, DeepLabCut, paralysie cérébrale unilatérale

AVANT-PROPOS

Ce travail de Bachelor de la filière Informatique de gestion a été réalisé à la Haute École Spécialisée de Suisse occidentale (HES-SO) Valais-Wallis à Sierre. Le sujet de ce rapport a été proposé par Madame Cristina Simon-Martinez, assistante post-doc à l'institut Informatique de gestion à Sierre. Le Prof. Dr Henning Müller, responsable de l'unité *eHealth* (santé digitale) à ce même institut, est le professeur chargé du suivi de ce rapport.

Le but poursuivi par ce travail est la réalisation d'un prototype de reconnaissance d'image afin de quantifier les mouvements en miroir sur des enfants atteints de paralysie cérébrale. Cela signifie qu'une réalisation technique pratique est effectuée conjointement à la rédaction pour développer un premier exemplaire prototype graphiquement.

J'ai choisi ce thème, car je souhaitais effectuer un travail orienté sur la pratique et pouvant être utile pour la suite au mandant. De plus, j'éprouve de la curiosité et un intérêt certain pour tout ce qui est lié à la reconnaissance d'image et l'apprentissage en profondeur par les systèmes d'information (*Deep Learning*). L'opportunité de travailler en collaboration avec des chercheurs pour la réalisation de ce travail m'a tout de suite séduit.

Ce rapport a été rédigé sur la base des normes de la 6^{e} édition du manuel de publication de *l'American Psychological Association* (APA). La forme de ce document est conforme à la version éditée en 2013 du guide de présentation et de réalisation des travaux rédigé par la Haute École de Gestion & Tourisme de la HES-SO Valais-Wallis. Après les remerciements, la première personne du pluriel sera privilégiée afin de respecter un cadre scientifique.

REMERCIEMENTS

Je tiens à remercier sincèrement les personnes qui ont contribué de près ou de loin à la réalisation de ce travail. Mes remerciements s'adressent plus particulièrement à :

- Monsieur Henning Müller, professeur responsable, pour le suivi régulier, les suggestions et les corrections de ce rapport ainsi que son soutien ;
- Madame Cristina Simon-Martinez, mandante et référente technique pour ce projet, pour ses explications et ses conseils avisés grâce à ses connaissances scientifiques et médicales dans le domaine des mouvements en miroir ;
- Madame Helga Haberfehlner, experte médicale externe, pour son écoute et ses recommandations sur la détection et la quantification des mouvements en miroir ;
- Mesdames Hilde Feys et Katrijn Klingels, qui m'ont permis d'exploiter des vidéos réalisées sur les mouvements en miroir dans le cadre d'une étude précédente ;
- Mes proches et mes amis pour leur soutien ainsi que la relecture de ce travail.

TABLE DES MATIÈRES

LISTES DES TABLEAUX

LISTES DES FIGURES

LISTE DES ABRÉVIATIONS

- 2D : Deux dimensions
- 3D : Trois dimensions
- aPCK : *adjusted Percentage of Correct Key points*
- APA : *American Psychological Association*
- API : *Application Programming Interface*
- AVX : *Advanced Vector Extensions*
- CNN : *Convolutional Neural Networks*
- CUDA : *Compute Unified Device Architecture*
- cuDNN : *CUDA Deep Neural Network library*
- Colab : Google Colaboratory
- CPU : *Central Processing Unit*
- CSV : *Comma-separated values*
- DenseNet : *Dense Convolutional Network*
- DLC : DeepLabCut
- ELT : *Extract Load Transform*
- ETL : *Extract Transform Load*
- GO : Gigaoctet
- GPU : *Graphics Processing Unit*
- GUI : *Graphical User Interface*
- HES-SO : Haute école spécialisée de Suisse occidentale
- IRMf : Imagerie par résonance magnétique fonctionnelle cérébrale
- maDLC : Multi-animal DeepLabCut
- MM : Mouvements en miroir
- OS : *Operating system*
- PC : Paralysie cérébrale
- RAM : *Random Access Memory*
- ResNet : *Residual Neural Network*
- RGPD : Règlement général sur la protection des données
- SFTP : *SSH File Transfer Protocol*
- SSH : *Secure Shell*
- TF : TensorFlow

GLOSSAIRE

- **Cloud** : Accès à des services informatiques, délocalisés auprès d'un fournisseur ou serveur externe, via internet.
- **Controlatéral** : Dont l'effet se manifeste du côté opposé au côté atteint.
- **Cortex moteur** : L'ensemble des aires du cortex cérébral qui participent à la planification, au contrôle et à l'exécution des mouvements volontaires des muscles du corps.
- **Inférence** : Capacité d'un modèle (*Deep* ou *Machine Learning*) entraîné à effectuer des prédictions sur des éléments donnés.
- **Inhibition** : Action nerveuse ou hormonale empêchant le bon fonctionnement d'un organe.
- **Ipsilatéral** : Relatif à quelque chose qui se produit ou se situe du même côté du corps.
- **Main parétique** : Main ipsilatérale à l'hémisphère sain.
- **Plasticité (cérébrale)** : La capacité du système nerveux central à modifier sa structure et ses connexions nerveuses au cours de la vie.
- **Réseaux de neurones convolutifs** : Sous-catégorie de réseaux de neurones conçus pour traiter des images en entrée.
- **Réseaux neuronaux résiduels** : Réseaux neuronaux artificiels basés sur des constructions obtenues à partir des cellules pyramidales du cortex cérébral.
- **Voie cortico-spinale** : Ensemble des fibres nerveuses directes ou indirectes unissant la circonvolution frontale ascendante (cortex moteur) à la moelle épinière.

INTRODUCTION

Contexte et problématique

Les mouvements en miroir (MM) sont des gestes involontaires d'une main qui reflètent les mouvements volontaires de l'autre main. Ils sont physiologiques pendant le développement de l'être humain (jusqu'à l'âge de 10 ans), mais peuvent subsister au-delà de cet âge après des lésions cérébrales, comme c'est le cas chez les enfants atteints de paralysie cérébrale unilatérale. Les mouvements en miroir peuvent affecter la façon dont ces enfants effectuent des activités bimanuelles, ce qui a un impact négatif sur leur qualité de vie (Klingels, et al., 2015, p. 735).

L'évaluation clinique des mouvements en miroir a été réalisée à l'aide d'une échelle permettant de quantifier leur sévérité (Woods & Teuber, 1978, p. 1153). Pour cette mesure, il est demandé à l'enfant d'effectuer trois tâches uni-manuelles (l'ouverture et la fermeture du poing, le lever des doigts et leur opposition), qui sont enregistrées à l'aide d'une caméra deux dimensions (2D) et notées ensuite sur une échelle ordinale allant de 0 (aucune activité en miroir) à 4 (activité en miroir égale). Cependant, cette appréciation reste subjective, car elle dépend de l'expérience de l'évaluateur.

Il y a eu plusieurs tentatives pour quantifier les mouvements en miroir avec des mesures basées sur la force (Zielinksi, et al., 2018; Jaspers, et al., 2017; Simon-Martinez, et al., 2020), qui fournissent une quantification continue de l'activité de miroir (similarité et intensité). Les inconvénients de ces évaluations sont qu'elles nécessitent des appareils et une expertise supplémentaire pour effectuer le test et les analyses en milieu clinique, ce qui augmente la charge de travail des équipes.

Les progrès de la vision par ordinateur (traiter et comprendre une ou plusieurs images prises par un système d'acquisition) et de l'analyse ce celles-ci ont rendu possible le développement de logiciels simples capables d'analyser et de suivre les mouvements dans une vidéo en 2D. Ces développements nous permettent donc de produire des images comme dans une situation classique lorsque nous filmons avec notre smartphone. Ceci se révèle être très utile en milieu clinique, car une telle acquisition d'images est facile à réaliser. Dans ce contexte, DeepLabCut (DLC) est une méthode efficace pour l'estimation de pose en 3D sans marqueur, basée sur l'apprentissage par transfert avec des réseaux neuronaux profonds (*deep neural networks*), qui permet d'obtenir d'excellents résultats (c'est-à-dire d'égaler la précision de l'étiquetage humain) avec des données d'entraînement minimales généralement comprises entre 50 et 200 images (Mathis, et al., 2018, p. 1281).

Objectifs

Alors que DeepLabCut a été initialement développé pour suivre le déplacement des animaux en laboratoire, ce projet vise désormais à transférer les connaissances acquises à la recherche clinique chez les enfants atteints de paralysie cérébrale unilatérale.

Le premier objectif est de développer un prototype de suivi de mouvements en 2D pour la quantification des mouvements en miroir chez les enfants atteints de paralysie cérébrale unilatérale et d'identifier les paramètres qui peuvent être utiles pour la compréhension clinique et neuroscientifique du phénomène des mouvements en miroir.

Le second objectif est de développer un logiciel pouvant être utilisé en milieu clinique afin de fournir aux spécialistes un outil permettant de télécharger une vidéo, d'estimer la pose et d'extraire les paramètres cliniquement importants pour quantifier et évaluer les mouvements en miroir. À terme, ce projet pourrait permettre de fournir un phénotype basé sur les MM qui pourrait aider à la prise de décision clinique.

Étapes

Pour introduire la thématique de ce travail, une revue de la littérature sera réalisée afin de se familiariser avec le phénomène des mouvements en miroir et le domaine de l'analyse vidéo par l'estimation de la pose.

Ensuite, une analyse des solutions de développement existantes sur le marché sera effectuée pour déterminer vers quels outils s'orienter. Un choix technologique comprenant l'environnement de développement et l'architecture du système d'information sera réalisé sur la base de cette analyse.

À l'aide d'un groupe de collaborateurs de l'institut de recherche en informatique de gestion et d'experts internationaux, des segments seront sélectionnés pour le suivi du mouvement. À la suite de cela, un prototype d'analyse vidéo pourra être réalisé avec DLC sur des vidéos d'enfants sains et d'autres avec des lésions cérébrales, afin d'extraire des paramètres différenciateurs pour quantifier les mouvements en miroir.

Dans le but d'aider les équipes cliniques à mesurer et quantifier les mouvements en miroir, une application expérimentale à l'aide d'une interface graphique sera créée et accessible dans un environnement de développement.

À la fin de ce travail, l'interface graphique pourrait être mise en phase de test pour les chercheurs et les cliniciens. Grâce à leurs remarques, les fonctionnalités développées vont pouvoir évoluer ou être éventuellement corrigées.

1. MOUVEMENTS EN MIROIR

1.1. Définition

Comme dit précédemment, les mouvements en miroir ou *mirror movements* sont des gestes involontaires d'un côté du corps qui reflètent des mouvements intentionnels du côté opposé (Galléa, et al., 2011, p. 1911). Ce comportement concerne principalement les membres supérieurs du corps humain (Galléa, et al., 2011, p. 1912).

Les MM apparaissent chez l'enfant et diminuent progressivement entre l'âge de cinq et huit ans. Dans la majorité des cas, ils disparaissent au-dessus de la dizaine d'années (Koerte, et al., 2010). Il n'est donc souvent pas alarmant de constater ceux-ci puisqu'ils font partie d'un processus normal du développement du système moteur chez l'enfant lorsqu'il grandit (Koerte, et al., 2010).

Malgré cela, les mouvements en miroir peuvent persister et être plus prononcés chez les enfants atteints de paralysie cérébrale (PC) unilatérale, et ce au-dessus de l'âge de 10 ans (Woods & Teuber, 1978, p. 1154; Connolly & Stratton, 1968, pp. 52-53).

1.2. Paralysie cérébrale unilatérale et système moteur

La PC est définie comme un trouble permanent du mouvement secondaire dû à une lésion cérébrale non progressive survenue soit pendant la période fœtale soit lors des deux premières années de vie (Dinomais, Hertz-Pannier, & Nguyen The Tich, 2014, p. 3). On parle d'unilatéralité pour qualifier une lésion qui concerne uniquement un hémisphère et dont les problèmes moteurs ne toucheront que la moitié du corps.

L'étude de Mailleux et al. (2021, p. 412) explique que la variabilité clinique des déficits des membres supérieurs chez les enfants atteints de paralysie cérébrale unilatérale est importante. Parmi ces enfants, certains présentent des problèmes subtils alors que d'autres ne peuvent même pas tenir un objet. Cette variabilité peut s'expliquer par les différentes caractéristiques neuroanatomiques d'une lésion cérébrale sous-jacente. Les lésions varient en fonction du moment, de la localisation ainsi que de leur étendue. Si elles surviennent lors du développement précoce du cerveau, elles peuvent modifier l'organisation de la principale

commande motrice, soit le tractus corticospinal et les projections sensorielles thalamocorticales[1].

Les technologies avancées en imagerie médicale et notamment l'imagerie par résonance magnétique fonctionnelle cérébrale (IRMf) ont permis d'en savoir plus sur le fonctionnement du cerveau. L'IRMf est une technique d'imagerie cérébrale permettant de mesurer *in vivo*[2] l'activité des aires du cerveau en détectant les changements locaux de flux sanguin (Andreeli & Mosbah, 2014, p. 13).

Grâce à la stimulation magnétique transcrânienne et à l'IRMf, les études réalisées ont permis de montrer que la plasticité[3] post-lésionnelle du contrôle central de la motricité (motricité cérébrale) pouvait se réaliser selon trois schémas (Dinomais, Hertz-Pannier, & Nguyen The Tich, 2014, p. 3; Mailleux, Simon-Martinez, Klingles, Ortibus, & Feys, 2021, p. 413).

Le premier schéma est la réorganisation ipsilatérale et concerne des lésions majeures du faisceau cortico-spinal. La main parétique[4] est contrôlée par le cortex moteur de l'hémisphère sain (Dinomais, et al., 2014, pp. 3-10). Le second schéma se retrouve chez des patients présentant un accident vasculaire cérébral néonatal de l'artère cérébrale moyenne et il s'agit alors d'une réorganisation controlatérale (Dinomais, et al., 2014, pp. 3-10). Quant au dernier *pattern*, il combine des projections ipsilatérales et controlatérales et on parle alors de réorganisation bilatérale (Staudt, 2010, p. 470).

Alors que les MM apparaissent chez l'enfant et se résolvent avec l'âge, les enfants atteints de PC unilatérale souffrent d'une activité miroir 15 fois plus importante, et ce, peu importe l'âge (Kuhtz-Buschbeck, Krumlinde-Sundholm, Eliasson, & Forssberg, 2007, p. 628).

L'étude des mouvements en miroir est importante pour améliorer la qualité de vie des enfants et adolescents qui en sont atteints. En effet, comme expliqué dans l'article de Klingels et al. (2015, p. 735), ceux-ci éprouvent souvent des difficultés de coordination bimanuelle, ce qui affecte leurs activités quotidiennes.

[1] Selon Sherman et Guillery, cela correspond aux principaux moteurs de l'activité corticale dans les zones sensorielles (cité dans Hunnicutt, et al., 2014, p. 1276).

[2] Dans l'organisme vivant.

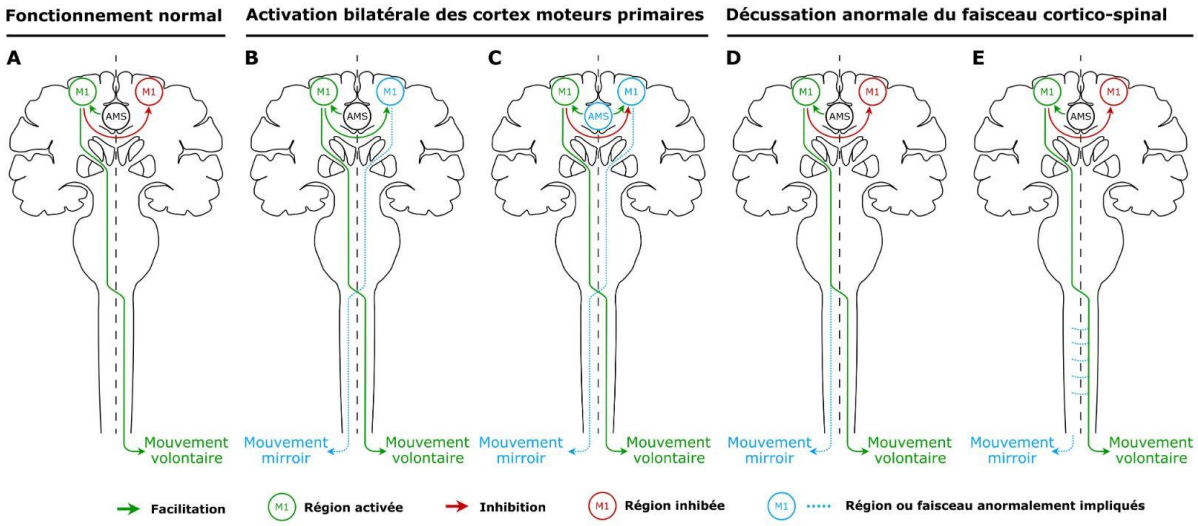
[3] La capacité du cerveau à créer ou à réorganiser des circuits neuronaux.

[4] Main ipsilatérale à l'hémisphère sain.

1.3. Causes et hypothèses

Les causes scientifiques du fonctionnement des mouvements en miroir dans l'organisme ne sont pas encore totalement comprises à ce jour, mais deux hypothèses se démarquent (Kuo, Friel, & Gordon, 2017, pp. 155-156). L'une d'elles pourrait être l'activation des cortex moteurs primaires bilatéraux (Carr, Harrison, Evans, & Stephens, 1993, p. 1237). Elle se manifesterait par une action nerveuse ou hormonale empêchant le bon fonctionnement d'un organe et serait causée par la lésion cérébrale. L'autre hypothèse serait la persistance de projections cortico-spinales ipsilatérales entre le cortex moteur non-lésionné et la main parétique (Koerte, et al., 2011, pp. 183-185).

Figure 1 : Représentation schématique des différentes hypothèses explicatives des MM



Source : Tisseyre, 2019, p. 27 adapté de Welnierz et al., 2015, p.3

La Figure 1 peut être complétée avec les explications suivantes :

(A) Dans des conditions normales, l'exécution d'un mouvement unimanuel de la main droite nécessite une activation du cortex moteur primaire (M1) controlatéral gauche, une inhibition interhémisphérique (IIH) du M1 ipsilatéral droit, une transmission du programme moteur approprié de l'aire motrice supplémentaire (AMS) vers le M1 gauche et enfin la transmission de la commande motrice vers les effecteurs de la main droite par le biais du faisceau cortico-spinal (FCS) croisé. Deux mécanismes principaux sous-tendent les MM : (1) une activation bilatérale des M1 résultant d'une facilitation interhémisphérique du M1 controlatéral sur le M1 ipsilatéral (B) ou d'une transmission anormale du programme moteur

de l'AMS vers les deux M1 (C) ; (2) une décussation anormale du FCS résultant d'une préservation du faisceau ipsilatéral non croisé (D) ou de ramifications aberrantes des axones du FCS au niveau de la moelle épinière (E) (Tisseyre, 2019, p. 27).

1.4. Diagnostics

L'échelle de Woods et Teuber (1978, p. 1153) est la plus utilisée pour détecter la présence de MM chez les enfants atteints de PC unilatérale (Magne, et al., 2021, p. 737). Celle-ci se base sur cinq niveaux distincts qui permettent ensuite d'attribuer un niveau de sévérité aux mouvements en miroir (Woods & Teuber, 1978, p. 1153). L'étude de Magne et al. (2021, p. 739) démontre que cette mesure fait preuve d'une excellente fiabilité comme évaluation des mouvements en miroir chez les enfants et les adolescents atteints de PC unilatérale.

Les trois tâches demandées pour procéder à une évaluation des MM selon Woods et Teuber (1978, p. 1153) sont les suivantes :

1. *Finger taping* : tapotement rapide de l'index sur l'articulation distale du pouce de la même main ;
2. *Fist tuning* : rotation du poing en alternant supination et pronation de l'avant-bras ;
3. *Finger alternation* : répétition de l'effleurement alterné répétitif de chaque doigt sur le bout du pouce de la même main, dans l'ordre suivant : pouce vers deuxième, troisième, quatrième, cinquième, deuxième, troisième, quatrième, cinquième, et ainsi de suite.

Bien qu'il s'agisse des trois tâches originelles, l'étude de Woods et Teuber datant de 1978 et ses tâches ont depuis été reprises et améliorées par d'autres études (Magne, et al., 2021, p. 737; Kuhtz-Buschbeck, Krumlinde-Sundholm, Eliasson, & Forssberg, 2007, p. 729). En effet, dans la pratique, la deuxième étape est parfois remplacée par une tâche qui consiste à ouvrir et fermer les doigts de la main de façon répétée pour observer comment la main opposée réagit. Une autre tâche commune est le tapotement des doigts sur une table (Magne, et al., 2021, p. 737).

Afin d'évaluer les MM dans les divers scénarios cités précédemment, il existe cinq scores selon Woods et Teuber (1978, p. 1153) :

0. Aucun mouvement imitatif clair ;
1. Mouvement répétitif à peine perceptible ;
2. Mouvement répétitif léger ou mouvement répétitif plus fort, mais plus bref ;

3. Mouvement répétitif fort et soutenu ;
4. Mouvement égal à celui attendu.

Le score est additionné pour chacune des trois tâches et les scores totaux possibles pour chaque main sont ainsi répartis de zéro à 12 (Woods & Teuber, 1978, p. 1153).

L'évaluation avec l'échelle de Woods et Teuber reste cependant subjective (Kuhtz-Buschbeck, Krumlinde-Sundholm, Eliasson, & Forssberg, 2007, p. 728; Magne, et al., 2021, p. 741), car il faut être expert ou avoir une formation préliminaire pour délivrer une appréciation correcte. Cette mesure est donc qualitative puisqu'elle ne se base pas sur des données. De plus, certains patients présentent une amplitude de mouvement limitée en raison de suractivité, raideur ou faiblesse musculaire et il peut être difficile de détecter les mouvements en miroir dans la main la plus affectée (Magne, et al., 2021, p. 741).

Certaines études se sont aussi penchées sur la mesure des mouvements en miroir par la force. L'une de ces méthodes est appelée *Windmill-task* et fournit une quantification objective des mouvements en miroir, mais nécessite un équipement technique qui n'est pas ou difficilement disponible cliniquement. Celle-ci fonctionne grâce à la récolte de données sur la force de préhension des deux mains pendant les mouvements de compression d'une seule main. Elle a été démontrée plus efficace que la façon de Woods et Teuber grâce à sa sensibilité et sa spécificité accrues (Zielinksi, et al., 2018, p. 1548). Malgré tout, la méthode initiale reste la plus populaire et la plus accessible puisqu'elle ne nécessite aucun équipement (Magne, et al., 2021, p. 737).

Comme les MM touchent majoritairement les membres supérieurs de l'être humain (Galléa, et al., 2011, p. 1912), la partie du corps la plus simple et intéressante à analyser dans ce travail serait les mains. En effet, celles-ci représentent un élément d'étude avantageux puisqu'elles constituent une surface raisonnable et sont facilement filmables. De plus, un nombre d'études conséquent ont également été réalisées sur la base de ce membre du corps humain (Mailleux, Simon-Martinez, Klingles, Ortibus, & Feys, 2021, p. 412; Klingels, et al., 2015, p. 737; Zielinksi, et al., 2018, p. 1549).

Avec les développements digitaux des dernières années, de nouvelles technologies sont apparues notamment dans le domaine de la reconnaissance d'image. Ce travail vise à utiliser l'analyse de vidéos grâce à l'estimation de la pose. Cette technique est rendue facile puisqu'elle ne nécessite plus d'équipement spécifique et peut donc être accessible en milieu clinique.

2. ESTIMATION DE LA POSE

La vision par ordinateur (*computer vision*) est une catégorie d'intelligence artificielle qui traite la modélisation de la vision humaine. Parmi les sous-catégories de la vision par ordinateur figure la classification et la segmentation des images, la détection des objets ou des visages et l'estimation de la pose (Odemakinde, 2021).

Cette dernière analyse la pose et l'orientation pour prédire et suivre l'emplacement d'une personne ou d'un objet. Elle permet d'estimer les positions spatiales d'un corps dans une image ou une vidéo. Pour ce faire, elle utilise les points clés d'un objet ou d'une personne, par exemple pour une personne, les articulations telles que le coude, les genoux et les poignets seront utilisées comme points clés. Il est également possible d'utiliser le terme *multi-pose* lorsque l'on se concentre sur plusieurs objets différents (Odemakinde, 2021).

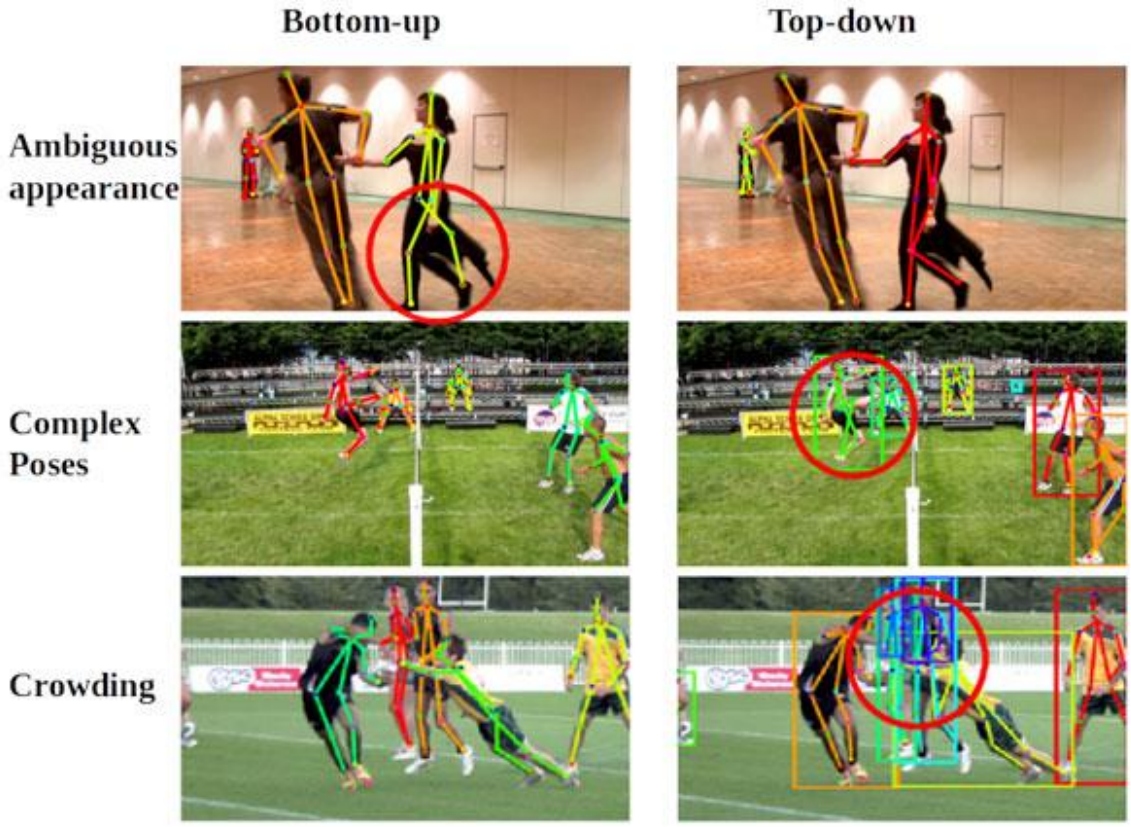
L'estimation de la pose fait donc partie du domaine de la reconnaissance d'image par ordinateur. Celle-ci va particulièrement être intéressante dans le cadre de ce travail pour estimer les mouvements des mains.

2.1. Architectures

Il existe plusieurs types d'architecture pour l'estimation de pose personnalisée. Parmi les plus connues, on retrouve les techniques suivantes : High Resolution Net (HRNet), OpenPose, DeepCut, Regional Multi-Person Pose Estimation (AlphaPose), DeepPose, PoseNet et DensePose.

En général, les architectures d'apprentissage en profondeur adaptées à l'estimation de pose sont basées sur des variations de réseaux de neurones convolutifs (CNN). Il existe deux approches globales : une approche ascendante et une approche descendante (Fritz Labs Incorporated, 2021).

Figure 2 : Comparaison des approches ascendantes et descendantes



Source : Fritz Labs Incorporated, 2021

Avec une approche ascendante ou *bottom-up* (Figure 2), le modèle détecte chaque instance d'un point clé particulier (par exemple, tous les pieds gauches) dans une image donnée, puis tente d'assembler des groupes de points clés en squelettes pour des objets distincts (Fritz Labs Incorporated, 2021). AlphaPose et HRNet font partie des architectures se basant sur cette méthode.

Une approche descendante ou *top-down* (Figure 2) fonctionne à l'inverse puisque le réseau utilise d'abord un détecteur d'objets pour dessiner une boîte autour de chaque instance d'un objet, puis estime les points clés dans chaque région recadrée (Fritz Labs Incorporated, 2021). Parmi les techniques connues qui adoptent cette méthode, on retrouve DeepCut et OpenPose.

3. MÉTHODES

Dans ce chapitre, nous allons analyser et comparer les différents logiciels existants sur le marché tant au niveau des caractéristiques, des performances et de la capacité de détection des mouvements. Les contraintes liées à la réalisation du prototype seront ensuite exposées.

3.1. Détection des mouvements

Tout d'abord, une sélection de programmes remplissant les conditions minimales sera effectuée. Ensuite, une analyse comparative sera réalisée afin de définir les différents critères présents sur chaque programme. Par la suite, une évaluation quantitative permettra de présenter leur efficacité.

Un choix technologique découlera de ces approfondissements pour s'orienter vers une solution en particulier. Une présentation et des explications sur le logiciel seront fournies pour comprendre son fonctionnement. Pour terminer, nous éclaircirons les façons dont il peut être exploité et les enjeux qui en dépendent.

3.1.1. Méthodes et logiciels existants

La détection des mouvements s'effectue grâce à l'estimation de la pose. Il existe différents logiciels permettant d'effectuer cette tâche avec des spécificités et algorithmes différents. On retrouve dans les fonctionnalités la capacité de reconnaissance d'image en 2D, en 3D ou encore le *live tracking* (estimation en direct). Certains programmes ne permettent pas d'analyser la posture de toutes les espèces d'animaux ou d'humains. C'est un aspect à prendre en compte lors des recherches d'un logiciel performant et correspondant au besoin de la tâche à réaliser.

Dans ce travail, nous devons disposer d'un logiciel répondant aux critères suivants : estimation de la pose en 2D, possibilité de suivre le mouvement humain et analyse vidéo. Lors de l'examen de ces prérequis, les solutions informatiques suivantes correspondaient aux critères : DeepBehaviour, DeepLabCut, DeepPoseKit, Optiflex et SLEAP (successeur de LEAP).

3.1.2. Analyse comparative

Dans cette analyse comparative, nous évaluons les logiciels retenus au sous-chapitre précédent. Nous allons examiner le tableau (Annexe I) mis en place afin de classer les différents critères de chaque programme. Cela nous permet de mettre en évidence les divers services et les capacités offerts.

Tout d'abord, nous constatons que seuls deux outils (DeepLabCut et SLEAP) possèdent une interface graphique complète connue sous le nom de *Graphical User Interface* (GUI). Les autres systèmes ne proposent que partiellement cette fonctionnalité et cela signifie qu'il faudrait paramétrer et exécuter les tâches en ligne de commande. Le plus important est que la labellisation des images soit possible avec une interface graphique, ce qui relèverait d'une tâche bien trop fastidieuse en ligne de commande. Ce critère n'est pas éliminatoire, mais lors de la prise en main d'un logiciel, il est souvent plus évident d'avoir un GUI à disposition pour réaliser rapidement un prototype et comprendre le processus dans son ensemble.

Nous notons également que tous les logiciels proposent des données d'exemples excepté DeepBehaviour. Ces dernières sont importantes pour tester si l'installation s'est déroulée correctement et s'assurer que le programme soit fonctionnel. Elles servent aussi à analyser le code d'un projet existant pour en comprendre le fonctionnement.

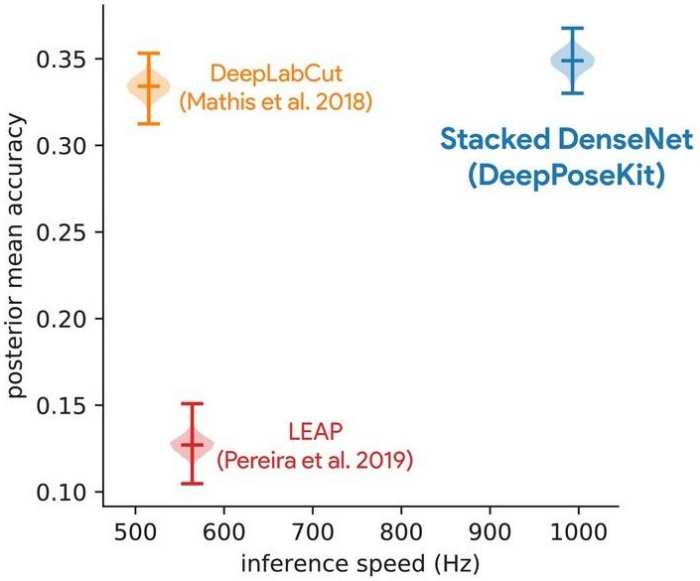
Ensuite, la documentation et la communauté d'utilisateurs représentent des critères pertinents et importants lors de l'utilisation d'un *software*. En effet, elles permettent à un utilisateur de pouvoir chercher des réponses à ses questions ou d'apprendre rapidement à acquérir les compétences nécessaires pour employer la technologie correctement. Nous remarquons qu'un logiciel, DeepLabCut, sort particulièrement du lot puisqu'il propose de multiples guides d'utilisation, un *workshop* et est doté d'une communauté étoffée avec un forum, un *chat* et un contact direct avec les développeurs ou les utilisateurs sur la plateforme GitHub.

Pour finir, le nombre de citations permettant d'évaluer la popularité des outils est récupéré grâce à Google Scholar. Il ne prend en considération que l'article le plus cité se référant au logiciel en question. En termes de popularité, nous constatons que DeepLabCut se démarque de nouveau avec plus de 838 citations dans d'autres articles scientifiques.

3.1.3. Analyses quantitatives

Afin d'évaluer les performances de certains estimateurs de pose retenus précédemment (point 3.1.2), nous allons reprendre des représentations quantitatives d'études déjà réalisées et les analyser. Dans un premier temps, nous examinerons des graphiques retenus d'un rapport rédigé sur DeepPoseKit, puis nous réitérerons ce procédé pour un article plus récent écrit sur OptiFlex.

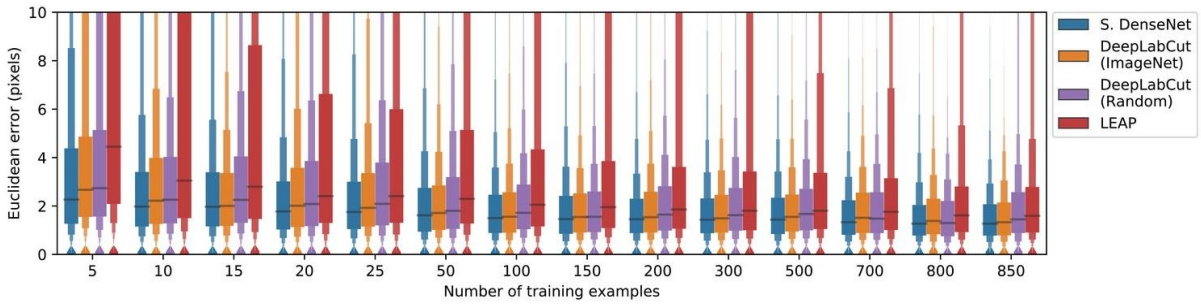
Figure 3 : Précision moyenne de plusieurs algorithmes



Source : Graving, et al., 2019, p.11

Des chercheurs de DeepPoseKit ont comparé la précision moyenne et la vitesse d'inférence sur un ensemble de données constitué de multiples zèbres sauvages en interaction (Figure 3). On remarque que DeepLabCut et DeepPoseKit, par son modèle nommé Stacked DenseNet, parviennent à une précision moyenne similaire avec un avantage léger pour le second. LEAP devenu aujourd'hui SLEAP est très imprécis pour cet ensemble de données complexes et son manque de précision est important. Nous constatons que DeepPoseKit surpasse ses deux autres concurrents puisqu'il est environ deux fois plus véloce qu'eux avec sa vitesse d'inférence (Graving, et al., 2019, p. 11). Celle-ci est définie par la rapidité d'un *software* à exécuter son algorithme sur des données nouvelles, soit des éléments qui n'ont pas été utilisés pour entraîner le modèle en question. Pour une meilleure compréhension, un schéma relatif à l'inférence est disponible (Annexe II).

Figure 4 : Comparaison de la précision de prédiction des algorithmes



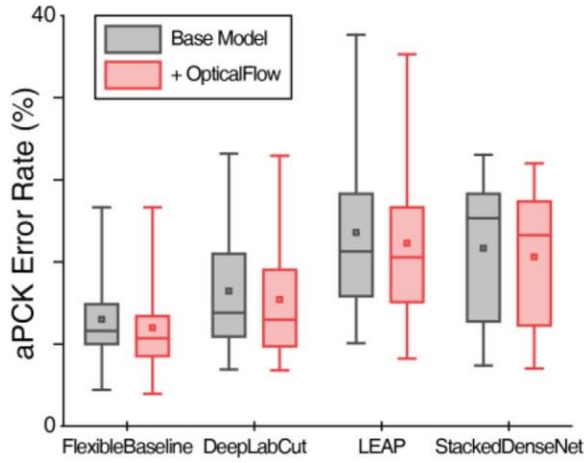
Source : Graving, et al., 2019, p.28

Un critère important à évaluer est la capacité d'un programme à être performant selon le nombre d'exemples d'entraînement que nous allons lui donner. Trois logiciels sont comparés avec différents nombres d'échantillons d'entraînement sur des données de zèbres (Figure 4), dont un avec deux dérivations d'algorithmes[5]. Le premier axe évalue le nombre d'exemples d'entraînements donnés en entrée (*input*). Le deuxième axe représente la distance euclidienne en pixel entre le point réel et le point estimé par une des solutions informatiques.

Le modèle *Stacked DenseNet* se démarque de nouveau. Il demeure plus efficace que ses concurrents, et ce, même avec un petit nombre de données d'exercice. En revanche, plus le nombre d'exemples est élevé, plus son efficacité diminue face à ses compétiteurs. Une légère différence entre le leader et les deux dérivés de DeepLabCut se fait remarquer, mais le retard majeur de LEAP jusqu'à 20 données d'apprentissage attire particulièrement l'attention. Au-delà de ce nombre, ce recul se résorbe et devient moins important.

[5] ImageNet qui est pré-entraîné et *Random* qui ne l'est pas

Figure 5 : Comparaison de performance de reconnaissance de points



Source : Liu, et al., 2021, p.7

Après avoir comparé deux graphiques de DeepPoseKit, nous allons nous pencher sur une récente étude menée par des chercheurs de OptiFlex. Des boîtes à moustaches (Figure 5) comparent la capacité de détection de différents algorithmes, dont FlexibleBaseline qui est le modèle racine qu'utilise OptiFlex. Il s'agit d'un cas d'application particulier, puisque la détection se fait sur la vue latérale d'une souris.

Nous remarquons que le modèle d'Optiflex est le plus performant, suivi de celui de DeepLabCut. Quant à LEAP et Stacked DenseNet, ils sont moins performants pour cette tâche avec une imprécision et une variabilité forte pour LEAP, qui est marquée dans le graphique par une moustache supérieure longue (distance entre le troisième quartile et le maximum).

De plus, nous constatons qu'il n'y a pas seulement les modèles de base qui sont comparés, mais aussi une modification apportée dans leur algorithme à l'aide du module OpticalFlow, représentée par les moustaches rouges. Celui-ci contient un procédé permettant que, même lorsque certains points clés ne sont pas visibles dans l'image cible, les informations transformées des images voisines fournissent toujours des critères suffisants sur l'emplacement le plus probable des points clés pour l'image cible (Liu, et al., 2021, p. 4).

Dans tous les cas, le module OpticalFlow est capable d'améliorer le modèle de base de chaque algorithme. Nous sommes en mesure de constater que toutes les moustaches rouges comparées aux grises abaissent légèrement le pourcentage d'erreur.

L'étude de Liu et al. (2021, p. 9) met aussi en lumière les capacités de détection de points clés des modèles (Annexe III) sur différents animaux et un objet. Nous observons que, dans l'ensemble des cinq graphiques, le meilleur modèle reste FlexibleBaseline. Ce dernier est suivi

de près par DeepLabCut, dont le taux d'erreur est bas. LEAP et Stacked DenseNet se partagent la troisième place avec des imprécisions notables dans les deux camps.

Avant de conclure ces analyses quantitatives, nous devons conserver un avis critique sur ce qui a été dit dans ce point. En effet, les deux études ont été menées par des chercheurs qui ont développé le logiciel ou modèle d'estimation de pose avec lequel ils se comparent aux autres. La première, relative à DeepPoseKit, a été réalisée avec un ensemble de données complexes et se focalise sur ce même programme. La deuxième se concentre dans un premier temps sur sa spécialité, qui est la vue latérale, mais bénéficie ensuite d'une comparaison plus objective à l'aide de graphique sur divers animaux et sur un fruit.

Finalement, nous décelons dans ces deux publications scientifiques que DeepLabCut obtient des résultats intéressants et se positionne parmi les leaders. Ce solide bilan vient du fait que DLC semble bien s'adapter à plusieurs types de situations. LEAP affiche, quant à lui, les moins bonnes performances. Cela ne signifie pas pour autant que le modèle n'est pas efficace, mais qu'il a ses défauts. OptiFlex et DeepPoseKit possèdent tous les deux leurs avantages dans des terrains spécifiques et semblent être des solutions performantes, mais qui disposent de peu de documentation en comparaison avec DeepLabCut. DeepBehaviour ne contient pas suffisamment de données statistiques pour être évalué dans cette analyse.

3.1.4. Choix technologique

L'utilisation de DLC est imposée dans la donnée de ce rapport, mais nous sommes en droit de nous demander s'il s'agit d'une solution adéquate. Après l'analyse comparative, il se révèle être un choix pertinent. En effet, les capacités techniques et les supports mis à disposition sont qualitatifs et aboutis. Cela permet à l'utilisateur d'être autonome et efficace. De plus, les analyses quantitatives montrent la polyvalence de DeepLabCut sur plusieurs cas d'utilisation.

Grâce à son interface graphique, le logiciel donne la possibilité d'extraire des *frames* (images) à partir d'une ou plusieurs vidéos puis de les labéliser. Cela nous permet d'obtenir une quantité intéressante d'images qui serviront de modèle pour entraîner le *software* à reconnaître plus tard les différents points dans une vidéo. Ses mises à jour récurrentes et ses performances font de lui un acteur de confiance et de fiabilité dans l'estimation de pose.

3.1.5. DeepLabCut, un combiné d'algorithmes

Après avoir cerné les avantages et les performances de DLC dans les points précédents, son fonctionnement peut désormais être approfondi.

DeepLabCut est un réseau convolutif profond qui combine deux éléments clés des algorithmes de reconnaissance d'objets et de segmentation sémantique : des réseaux neuronaux résiduels (ResNets) pré-entraînés et des couches convolutives transposées (Mathis, et al., 2018, p. 1282).

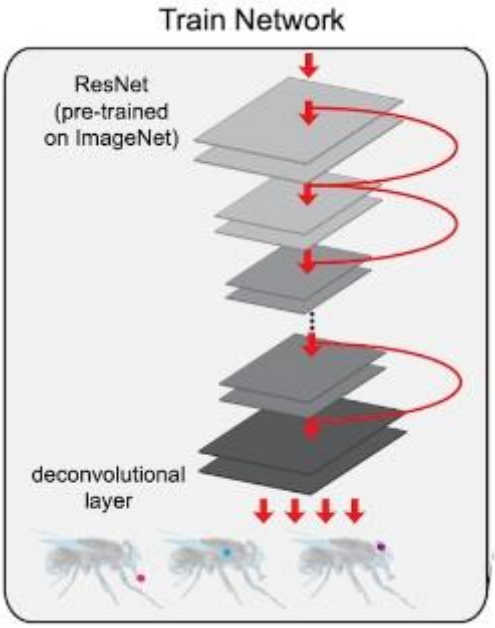
> La segmentation sémantique est un algorithme de *Deep Learning* qui associe une étiquette ou une catégorie à chaque pixel d'une image. Elle permet de reconnaître un ensemble de pixels qui forment des catégories distinctes. Par exemple, un véhicule autonome doit identifier des véhicules, des piétons, des panneaux de signalisation, des trottoirs et autres éléments de l'environnement routier (The MathWorks, Inc., s.d.).

La segmentation sémantique est très performante en termes de cartographie des images. Elle est utilisée pour distinguer des éléments précis, par exemple, sur une carte géographique, les rivières des montagnes et les montagnes des forêts.

Le réseau convolutif de DeepLabCut est constitué d'une variante de ResNets, dont les poids ont été entraînés sur un repère populaire de reconnaissance d'objets à grande échelle appelé ImageNet, sur lequel il obtient d'excellentes performances (Mathis, et al., 2018, p. 1282).

> *Instead of the classification layer at the output of the ResNet, deconvolutional layers are used to up-sample the visual information and produce spatial probability densities. For each body part, its probability density represents the 'evidence' that a body part is in a particular location. To fine-tune the network for a particular task, its weights are trained on labeled data, which consist of frames and the accompanying annotated body part locations (or other objects of interest in the frame)* (Mathis, et al., 2018, p. 1282).

Figure 6 : Schématisation de l'entraînement d'un réseau avec DeepLabCut



Source : Nath et al. (2019)

Le schéma (Figure 6) montre les étapes de fonctionnement de DeepLabCut en partant des images pré-entraînées jusqu'au *deconvolutional layer.* Au fur et à mesure de l'entraînement, le logiciel attribue des probabilités fortes ou basses pour chaque point labélisé.

> *During training, the weights are adjusted in an iterative fashion such that for a given frame the network assigns high probabilities to labeled body part locations and low probabilities elsewhere…. Thereby, the network is rewired and 'learns' feature detectors for the labeled body parts. As a result of the initialization with the ResNet pretrained on ImageNet, this rewiring is robust and data-efficient* (Mathis, et al., 2018, p. 1282).

3.1.6. Processeurs

DLC offre la possibilité de choisir avec quel type de processeur nous souhaitons travailler. Cependant, une des deux solutions semble radicalement plus intéressante. Premièrement, il est possible d'utiliser l'unité centrale de traitement plus communément appelée *Central Processing Unit* (CPU) ou simplement processeur. Celui-ci agit au cœur de l'ordinateur et exécute les programmes (Mathis, Schneider, Lauer, & Mathis, 2020, p. 55).

La deuxième solution proposée est de se servir de l'unité de traitement graphique que l'on connaît sous le nom de *Graphics Processing Unit* (GPU) ou encore processeur graphique.

Nous parlons ici d'un dispositif informatique spécialisé, conçu pour traiter et modifier rapidement la mémoire (Mathis, Schneider, Lauer, & Mathis, 2020, p. 55).

Le processeur graphique peut être intégré par certains fabricants directement avec le CPU selon le principe de mémoire partagée. À ce moment-là, le processeur graphique utilisera la mémoire partagée avec l'unité centrale de traitement. Néanmoins, la plupart du temps, on retrouve le GPU dans une carte graphique, car il sera indépendant, plus puissant et donc généralement plus performant que lorsqu'il est intégré directement au processeur.

En utilisant DLC, c'est la mémoire d'un processeur qui va être sollicitée. Celle-ci devra avoir une capacité minimum de huit gigaoctets (GO) pour une mémoire graphique ou 32 GO pour la *Random Access Memory* (RAM) pour travailler avec un CPU (Tanmay, et al., 2019).

Dans ce travail, nous privilégions un processeur graphique. Effectivement, il est de 10 à 100 fois plus rapide que les CPU à six cœurs (Mathis & Warren, 2018, p. 8).

3.1.7. Solution *cloud*

Lors de l'utilisation de DeepLabCut, la puissance d'un processeur classique ou graphique est demandée. Pour ce faire, elle requiert des éléments physiques (*hardware*) intégrés à un ordinateur ou un serveur. Grâce aux avancées des services de *cloud computing* et à la diversification de l'offre, il existe des fournisseurs qui proposent de la puissance de processeur à distance.

C'est le cas de Google qui a mis en place sa solution gratuite Google Colaboratory[6] souvent raccourcie en « Colab ». Elle permet l'écriture et l'exécution du code de programmation Python dans un navigateur. Destiné aux étudiants, aux *data scientist* ou aux chercheurs en intelligence artificielle, ce service simplifie grandement l'exécution des tâches grâce à plusieurs GPU s'exécutant sur les serveurs Google (Google, s.d.). Il existe aussi le fournisseur en accès libre MyBinder[7] qui permet d'exécuter directement du code provenant d'un *repository* Git, soit un emplacement de stockage en ligne pour déposer son code. Son principal désavantage est qu'il ne propose pas d'unité de traitement graphique et il ne peut donc pas être envisagé dans notre contexte. D'autres solutions payantes existent avec des exécutions effectuées dans le *cloud*.

[6] Disponible sur https://colab.research.google.com

[7] Disponible sur https://mybinder.org/

Les deux solutions gratuites mentionnées ci-dessus sont basées sur le *notebook* Jupyter, « *an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more* » (Project Jupyter, 2021).

DLC a établi plusieurs documentations et des exemples sur l'utilisation de son code avec Colab. Pour cette raison, Colab sera préféré dans ce travail, car son fonctionnement et sa compatibilité sont déjà prouvés par l'éditeur du logiciel.

3.2. Contraintes pour le prototype

Lorsque nous aurons effectué la détection et la quantification, l'objectif sera de rendre l'utilisation de notre modèle exécutable sur une interface graphique pour les cliniciens. Afin de réaliser le prototype, il faut définir quelle forme prendra notre programme. Après, nous devrons comprendre quelles infrastructures sont nécessaires pour afficher les données et procéder à leur traitement.

3.2.1. Sélection technologique

Il est nécessaire que le prototype soit facile d'utilisation et accessible. Le choix se porte sur l'un des trois types de logiciels suivants :

1. Une interface graphique de type GUI *standalone*, sans connexion internet nécessaire ;
2. Un GUI à installer en local avec une communication sur le web ;
3. Une application web.

La première option n'est pas intéressante puisqu'elle a un désavantage majeur. Celui-ci est la compatibilité du *software* pour chaque *operating system* (OS) distinct ainsi que les versions diverses. En effet, il est très difficile de rendre une interface graphique multi-OS à cause de la complexité de codage d'un programme dans différents langages. De plus, cette option n'est pas réalisable, car nous souhaitons avoir de la puissance graphique pour traiter rapidement le processus d'inférence. Cela signifie que les machines souhaitant utiliser le logiciel devraient être équipées d'un GPU. D'autre part, la rapidité d'exécution doit aussi être prise en compte puisqu'il sera nécessaire d'installer en local le *software* et cela peut décourager les utilisateurs. Non seulement ils devront procéder à une installation, mais ils n'auront peut-être pas les droits d'administrateurs pour exécuter cette action.

La seconde alternative pourrait éliminer le souci matériel, car elle permettrait d'aller chercher les ressources nécessaires dans le *cloud* ou sur un serveur. Cependant, elle ne résoudra pas le problème de la compatibilité entre les différents systèmes d'exploitation.

En revanche, la dernière éventualité pourrait être intéressante puisqu'en construisant une application web, elle serait utilisable sur tous les OS. Pourtant, un obstacle vient s'ajouter, car ce sont les navigateurs qui pourraient ne pas supporter l'application. Plus précisément, c'est le *framework* avec lequel le développement aura lieu qui déterminera le fonctionnement ou non de notre application dans les *browsers*. Malgré tout, la majorité des cliniques sont aujourd'hui reliées à internet et c'est pour cette raison qu'opter pour une application web est la solution la plus pratique et la plus accessible. C'est donc cette alternative qui sera privilégiée pour le développement dans ce travail.

3.2.2. Backend

Une application web contient une partie visible accessible à l'utilisateur (*frontend*) et une partie invisible (*backend*) où tous les services et tâches sont effectués. L'appel à un environnement d'exécution pour produire le processus d'inférence sera donc réalisé par ce dernier.

Tout d'abord, la création et la mise en place d'une base de données ne semblent pas pertinentes dans ce travail, car nous ne souhaitons pas stocker d'informations. Effectivement, le fait que les vidéos traitées par notre application web proviennent de milieux cliniques rend primordial d'assurer la confidentialité du service que nous souhaitons proposer. Certes, récolter des données serait intéressant pour faire évoluer notre application, mais celles-ci appartiennent au domaine médical et restent donc sensibles.

Une exécution décentralisée est essentielle pour fournir la mémoire graphique qui accélère considérablement le temps d'inférence. Les applications web classiques sont hébergées sur des serveurs basiques qui sont généralement équipés d'un CPU, mais sans unité de traitement graphique dédiée.

Précédemment, nous avons travaillé avec Google Colaboratory, qui est une application web basée sur le notebook Jupyter. Cette solution n'a pas été conçue pour agir en tant que *backend* pour une interface graphique, mais uniquement pour un usage sur sa plateforme. La seule possibilité pour réaliser une telle opération serait de connecter Colab à un serveur qui agirait comme *frontend*[8]. En apparence, cela semble simple et optimal, mais il faut avoir deux

[8] Example de réalisation sur https://anvil.works/learn/tutorials/google-colab-to-web-app

fenêtres de navigateur ouvertes, lancer manuellement la connexion dans le *notebook* et une fois celle-ci accomplie, l'interface graphique peut travailler avec le backend. De plus, Google n'a pas créé ce service à cette fin, mais dans le but qu'il soit accessible gratuitement au plus grand nombre de *data scientist,* chercheurs en intelligence artificielle et autres étudiants à des fins de recherches (Google, s.d.).

Ainsi, pour disposer de mémoire graphique, plusieurs choix sont possibles, comme la mise en place d'un serveur d'exécution physique ou virtuel (*cloud* GPU). Il peut être celui qui héberge l'application web ou alors être contacté pour effectuer seulement l'exécution de l'inférence par une *Application Programming Interface* (API). Aucune documentation sur l'achèvement d'une telle application web d'inférence avec DeepLabCut n'a été trouvée et il faudra donc explorer les diverses architectures possibles.

3.2.3. Frontend

La partie visible de l'iceberg de l'application, soit le *frontend*, peut être envisagée sous différents formats. Parmi eux, la possibilité de reprendre ce qui a été fait dans Colaboratory pour l'exporter dans un *notebook* Jupyter ou réaliser une interface graphique indépendante à l'aide d'un *framework.*

Comme le processus d'analyse et d'estimation de pose d'une vidéo a déjà été réalisé sur Colab dans un carnet, nous pourrions essayer d'en tirer parti. Pour accomplir ceci, il faudrait transporter l'existant dans une interface de *notebook* Jupyter indépendante. À l'aide d'un moteur graphique qui emploie des éléments interactifs, il serait possible de créer une application web.

D'autre part, un environnement de développement pourrait aussi être choisi, soit directement en Python avec un *framework* comme Django ou Flask, soit avec d'autres infrastructures logicielles par exemple React ou Vue.js (JavaScript), Angular (TypeScript) ou encore Laravel (PHP). Le choix du *frontend* se fera sur la base de ce qui a déjà été réalisé dans ce rapport ainsi qu'avec une analyse de faisabilité.

4. RÉSULTATS

L'ensemble des moyens mis en œuvre pour réaliser un prototype graphique sera expliqué dans ce chapitre. D'abord, les techniques pour accomplir l'estimation de pose avec DeepLabCut seront choisies. Un modèle sera ensuite créé et entraîné afin d'être réutilisé pour l'analyse de vidéos d'inférence. Pour terminer, la quantification des mouvements en miroir sera traitée avant de construire l'interface graphique qui viendra lier tous les éléments de ce travail dans un processus d'exécution global.

4.1. Estimation de la pose avec DeepLabCut

Avant de créer un modèle, les enjeux doivent être analysés afin de définir comment l'estimation de la pose sera implémentée. Dans un premier temps, les prérequis pour installer DeepLabCut seront abordés avant l'installation de l'outil. Puis, un choix d'exercice à filmer sera nécessaire pour décider la manière dont les mouvements seront suivis puis analysés. Deux techniques de mise en œuvre de DLC seront ensuite comparées avant de les confronter dans une analyse de performance. Les points clés des mains seront traités pour communiquer au logiciel la façon dont le *tracking* doit être exécuté. Finalement, une amélioration éventuelle du prototype sera analysée.

4.1.1. Prérequis avant installation

Pour installer DeepLabCut, il est obligatoire d'avoir au minimum la version *3.7* de Python[9] installée. Pour utiliser le GPU, une carte graphique de la marque NVIDIA est obligatoire, car les autres ne sont pas supportées pour le moment. L'utilisateur doit disposer d'un pilote[10] (*driver*) pour son processeur graphique et doit encore installer la *Compute Unified Device Architecture* (CUDA[11]), technologie créée par l'un des plus grands fournisseurs de processeurs graphiques NVIDIA pour employer la puissance de calcul du GPU de cette même marque (DeepLabCut, 2021a).

Nous avons effectué l'installation avec une carte graphique « NVIDIA GeForce GTX 1660 Ti » pour prendre en main le logiciel et ses opérations d'exécution. Nous avons cependant préféré, plus tard, adopter Colaboratory de Google pour exécuter les opérations

[9] Disponible sur https://www.python.org/downloads/

[10] Disponible sur https://www.nvidia.com/Download/index.aspx

[11] Disponible sur https://developer.nvidia.com/cuda-downloads

nécessitant un GPU. En effet, sa portabilité est un atout majeur et a permis de faciliter la tâche, surtout pour la programmation d'un *notebook* réutilisable.

4.1.2. Installation pour Windows 10

DeepLabCut peut être installé en ligne de commande, mais nous recommandons l'installation et l'utilisation d'Anaconda,[12] car ce logiciel permet de créer des environnements de développements personnalisés. Il est très répandu dans la communauté des *data scientist*. Par conséquent, les éditeurs de programme proposent des fichiers de configuration pour installer l'environnement nécessaire à la bonne exécution de leur *software*.

Quatre étapes sont essentielles à l'installation de DLC avec Anaconda. La première consiste à télécharger ou cloner le *repository* GitHub[13] en local (DeepLabCut, 2021a).

La seconde phase comprend le lancement d'Anaconda Prompt. Il s'agit de l'interface en ligne de commande du logiciel. Il faut naviguer en mode terminal jusqu'à l'endroit physique où est stocké le fichier de paramétrage avec, par exemple, la commande ci-dessous (DeepLabCut, 2021a).

```
cd C:\Users\YourUserName\Desktop\DeepLabCut\conda-environments
```

Ensuite, une fois dans le dossier, il suffit de créer l'environnement à l'aide du fichier de configuration se trouvant dans le dossier « conda-environments » de DeepLabCut (DeepLabCut, 2021a).

```
conda env create -f DLC-GPU.yaml
```

Finalement, lorsque l'environnement est créé, il est possible de l'utiliser depuis n'importe quel endroit de notre machine. Pour l'activer et lancer le GUI, il suffit d'entrer les deux lignes de commandes ci-après (DeepLabCut, 2021a).

```
conda activate DLC-GPU
python -m deeplabcut
```

Les instructions détaillées sont disponibles sur le *repository* GitHub[14] de DLC. Nous avons travaillé dans ce rapport avec la version 2.2rc2 (DeepLabCut, 2021b).

[12] Disponible sur https://www.anaconda.com/products/individual

[13] Disponible sur https://github.com/DeepLabCut/DeepLabCut

[14] https://github.com/DeepLabCut/DeepLabCut/blob/master/docs/installation.md

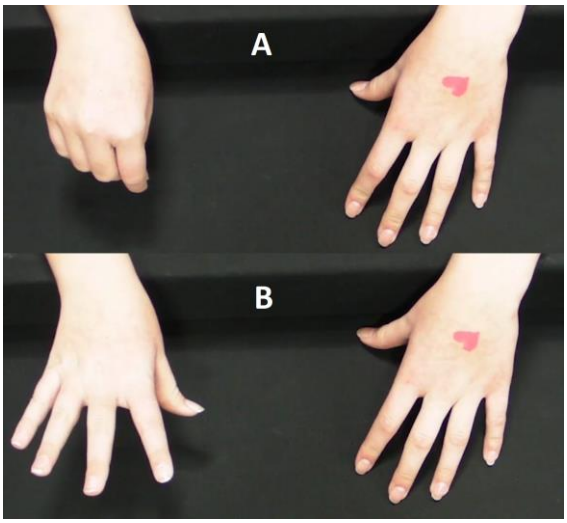
4.1.3. Choix de l'exercice

Afin d'effectuer l'estimation de la pose et de pouvoir quantifier les mouvements en miroir, il faut choisir une séquence intéressante à filmer. La complexité de la sélection d'un exercice adéquat réside en trois facteurs. Le premier est la manœuvre qui doit être reconnue médicalement comme étant valide pour l'évaluation des mouvements en miroir. Un autre élément qui influe sur ce choix est la capacité de *tracking* dont fera preuve le logiciel d'estimation de la pose. Le dernier élément pour obtenir un modèle bien entraîné est le nombre de vidéos à disposition pour un exercice. En effet, pour que l'algorithme soit performant, il faut lui donner matière à apprendre grâce à ces dernières.

Nous nous sommes penchés sur un article scientifique récent de Magne et al. (2021) qui contient six vidéos de bonne qualité dans lesquelles les mains sont clairement visibles. Pour une meilleure clarté dans cette étude, l'exercice se nommera « X » (Figure 7).

> *The assessment of mirror movements was performed with the participants seated comfortably at a table with the forearms resting on the table surface or at an elevated rim, providing space to move the hands freely. The participants were instructed to execute each repetitive task at a natural speed for 10 to 15 seconds with each hand individually, while the other hand was resting* (Magne, et al., 2021, p. 737).

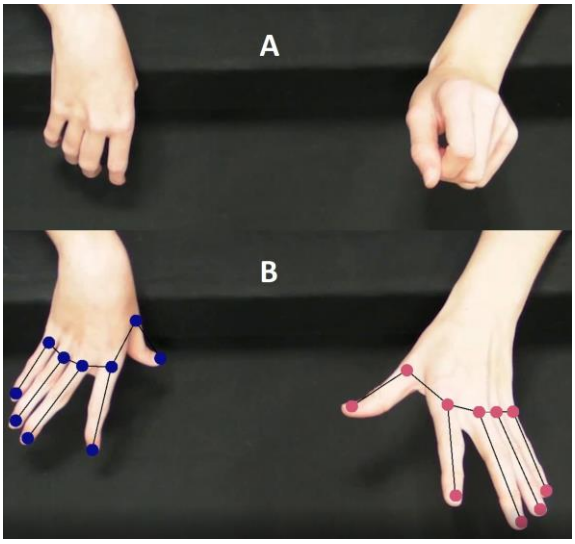
Figure 7 : Exercice X, une main en mouvement et l'autre au repos



Source : Magne et al. (2021)

Lors de la réalisation du tout premier essai avec DeepLabCut, nous avons travaillé avec ces vidéos (exemple Figure 7). Le but était de réaliser un modèle expérimental pour analyser d'une part comment le *tracking* apparaissait sur une vidéo analysée et d'autre part pour savoir comment étaient récoltées les données pour la quantification.

Figure 8 : Exercice X après analyse et création du suivi des mouvements



Source : Photo de l'auteur adaptée de Magne et al. (2021)

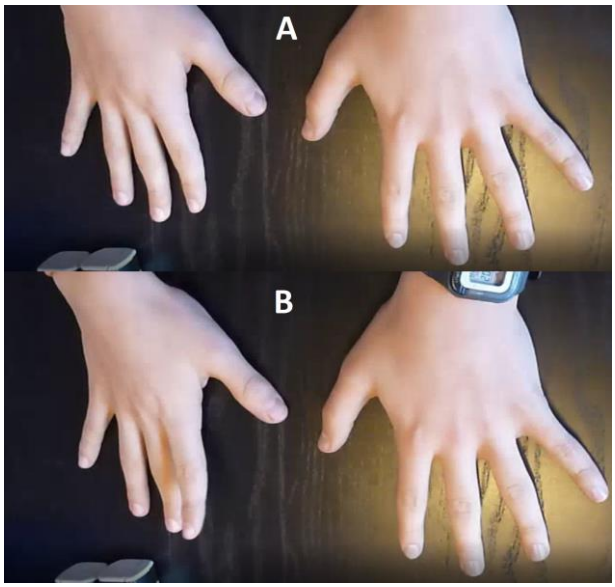
Le fait que les points clés comme le bout de l'index se retrouvent sous la main lors de cet exercice découle d'une occultation dans la vidéo (Figure 8). Cette dernière fait disparaître tous les repères lorsque les mains se ferment. En cause, cette première tentative n'a été que peu entraînée (2'000 itérations[15]) et l'outil n'est donc pas en mesure de prédire le mouvement. Cependant, étant donné que les disparitions sont régulières dans les images, entraîner un modèle capable de prévoir où se trouveront les points est une solution complexe. Cela nécessiterait un entraînement de masse ainsi que des résultats incertains et variables. De plus, les six vidéos se ressemblent, ce qui diminue le potentiel d'apprentissage du *software*.

Pour cette raison, nous avons entrepris d'autres recherches sur les exercices reconnus cliniquement. Parmi eux, la sélection s'est portée sur une manœuvre intéressante (appelée Y dans cette étude) qui consiste à poser les mains sur une table, lever et poser chaque doigt d'une main à tour de rôle alors que l'autre main reste au repos. Les Prof. Drs Katrijn Klingels et Hilde Feys ont donné leur accord pour mettre à disposition des vidéos où plusieurs exercices étaient filmés, dont celui cité précédemment. Ces dernières ne sont pas publiées et peuvent être utilisées dans ce rapport avec la condition *sine qua non* qu'elles soient anonymisées. Elles ont été utiles pour comprendre les déficits sensorimoteurs des membres supérieurs chez les

[15] Il est recommandé d'effectuer environ 500'000 itérations (DeepLabCut, 2021j).

enfants atteints de paralysie cérébrale unilatérale (Jaspers, Byblow, Feys, & Wenderoth, 2016, pp. 6-7; De Winter, Klingels, & Simon Martinez, 2016).

Figure 9 : Aperçu de l'exercice Y avec une main en mouvement et l'autre au repos



Source : Jaspers et al. (2016)

Contrairement à la figure précédente (Figure 8), la détection des mouvements de la Figure 9 par DeepLabCut n'implique pas ou peu d'occultations des différents points sur les images. Le fait que les vidéos aient été réalisées avec différents angles de vue ainsi que des arrière-plans variés est un avantage certain pour l'entraînement de notre modèle d'estimation de pose.

> *Having 10 videos that include different backgrounds, different individuals, and different postures is MUCH better than 1 or 2 videos of 1 or 2 different individuals (i.e. 10-20 frames from each of 10 videos is much better than 50-100 frames from 2 videos)* (DeepLabCut, 2021c).

Néanmoins, il faudra faire attention lors de l'inférence, car cela nécessite d'être précis pour que la quantification puisse se dérouler dans les meilleures conditions. Si l'angle de cadrage est trop haut lorsque la séquence est filmée (Figure 9), les mouvements seront minimisés et plus difficiles à calculer. Des directives précises quant aux vidéos destinées à l'inférence devront être rédigées afin d'expliquer les tenants et aboutissants aux cliniciens.

4.1.4. Standard et multi-animal

DLC propose deux manières distinctes de travailler avec, pour chacune, des spécificités à prendre en compte. La première, appelée *standard* DeepLabCut, effectue l'estimation de pose d'un ou plusieurs objets ou animaux à condition qu'ils soient aisément différenciables les uns des autres. La seconde, connue sous le nom de *multi-animal* DeepLabCut (maDLC), est destinée à la reconnaissance de plusieurs animaux ou objets similaires (DeepLabCut, 2021e).

« *For example, a white mouse + black mouse would call for standard, while two black mice would use multi-animal* » (DeepLabCut, 2021d).

Dans le cadre de ce rapport, les vidéos filmées contiennent uniquement les mains d'une personne sur une table ou toute autre surface plane. Bien que celles-ci se ressemblent et puissent être considérées comme deux éléments, puisqu'elles contiennent les mêmes points clés, les créateurs de DLC recommandent toujours de commencer avec un projet standard pour comprendre le *workflow* (DeepLabCut, 2021c).

L'approche standard pourrait suffire et est privilégiée pour le premier essai. Nous pensons que, du fait de la ressemblance importante entre les deux éléments, la piste maDLC peut être intéressante dans un deuxième temps. Le moyen décisionnel le plus efficace reste de réaliser deux prototypes et de les confronter.

4.1.5. Analyses de performances

Afin de réaliser deux prototypes équivalents, un projet standard et un multi-animal sont créés avec les mêmes vidéos labélisées. Les points clés et le nombre de fois que le modèle sera entraîné sont similaires.

Les valeurs par défaut de la taille de lot (batch size) sont d'un pour le *standard* DLC et de huit pour maDLC. C'est pourquoi le modèle normal contiendra deux entraînements, afin de produire des évaluations avec les mêmes configurations. Nous allons analyser l'impact que le *batch size* pourrait avoir.

Tableau 1 : Comparaison des modèles de type standard et multi-animal

Modèle	Batch size	Training iterations:	% Training dataset	Shuffle number	Train error (px)	Test error (px)	p-cutoff[16] used	Train error with p-cutoff	Test error with p-cutoff
DLC-1	1	20'000	95	1	4,08	4,16	0,60	4,08	4,16
DLC-1	8	20'000	95	1	3,81	4,07	0,60	3,80	4,07
maDLC-1	8	20'000	95	1	1,94	3,44	0,60	1,94	3,44

Source : Données de l'auteur

Pour 20'000 itérations, soit le nombre total de fois où notre prototype a été entraîné, nous remarquons, grâce au tableau ci-dessus (Tableau 1), une efficacité plus élevée pour détecter plusieurs animaux similaires. Les moins bons résultats appartiennent au modèle standard qui a une taille de lot d'un.

Le *batch size* définit le nombre d'échantillons à traiter avant de mettre à jour les paramètres du modèle interne qui est entraîné. À la fin d'un lot, les prédictions sont comparées aux variables de sortie attendues et un taux d'erreur est calculé. C'est à partir de celui-ci que l'algorithme de mise à jour est utilisé pour améliorer le modèle (Brownlee, 2018).

D'un point de vue objectif, cela n'a rien de surprenant que maDLC devance la version *standard,* car DeepLabCut utilise des algorithmes différents pour les différents types de projets. « *For multi-animal projects we are using not only different and new output layers, but also new data augmentation, optimization, learning rates, and batch training defaults* » (DeepLabCut, 2021e).

Effectivement, le code *multi-animal* est basé sur le programme standard de DLC et amélioré. Sa première version date du 3 mai 2021 et, depuis, l'équipe de développement a continué de l'améliorer (DeepLabCut, 2021b). Certes, ils n'ont pas abandonné l'utilisation *standard* du logiciel, mais elle était arrivée à maturité et nécessite désormais moins d'améliorations ou uniquement des correctifs mineurs.

Par ailleurs, le fait que la programmation soit différente pour les deux alternatives d'emploi du *software*, rend aussi le *multi-animal* plus lourd et plus lent à entraîner, mais cela est surtout causé par la taille de lot.

[16] Le *p-cutoff* correspond au taux de confidence pour la détection des différents points.

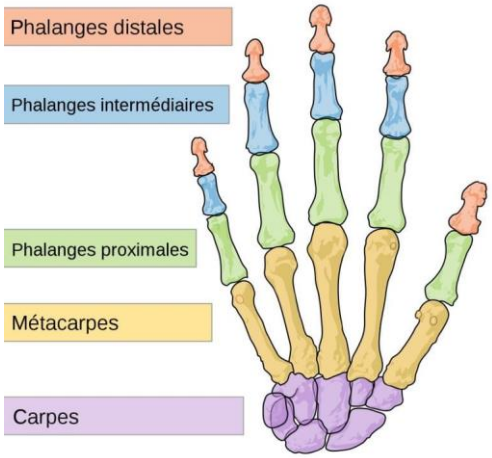
Le nombre d'itérations pour obtenir les meilleurs résultats dépend de la taille de lot et de la fonction DeepLabCut choisie. En version standard, avec un batch size d'un, il faut environ 500'000 itérations (DeepLabCut, 2021j). Si la taille est de huit, il faudrait 50'000 à 100'000 répétitions. Finalement, pour un maDLC, il faudrait entre 20'000 et 100'000 récurrences qui sont paramétrées par défaut avec un batch size de huit (DeepLabCut, 2021e).

De ce fait, l'analyse comparative effectuée ci-dessus reste généraliste puisque les paramètres diffèrent pour chaque type d'utilisation. Il est donc complexe de réaliser une évaluation au même niveau pour tous les modèles. Néanmoins, les résultats significatifs des bonnes performances du modèle *multi-animal* nous poussent à nous orienter vers ce modèle précis.

4.1.6. Points clés d'une main

Pour définir les points clés d'une main (aussi appelé labels lors de l'utilisation dans DLC), il est primordial de se poser les bonnes questions. D'abord, nous devons connaître les points clés essentiels à la réalisation de notre but. Ensuite, la lecture de la littérature peut apporter des réponses à notre problème à l'aide de modèles existants.

Figure 10 : Illustration d'ossature d'une main



Source : TOOMEDICAL (2020)

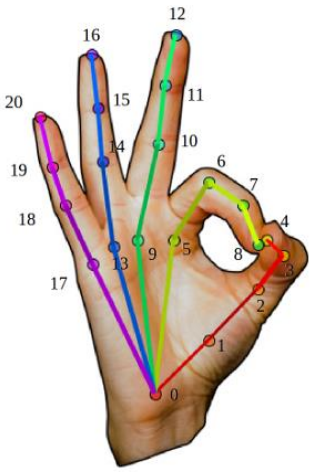
L'essentiel pour ce rapport est d'obtenir les mouvements des différents doigts d'une main pour les comparer à l'autre. Cependant, donner au logiciel uniquement les phalanges distales (Figure 10) ne suffit pas, car il a besoin de plus de données pour effectuer l'estimation de pose.

Figure 11 : Premier prototype standard DLC-1 avec labels

Source : Photo de l'auteur adaptée de Jaspers et al. (2016)

Dans les premiers prototypes réalisés (Figure 11) et analysés précédemment, les points clés étaient les phalanges distales ainsi que la jonction entre les phalanges proximales et le métacarpe. Cette façon de procéder pouvait sembler attrayante afin d'obtenir les premiers résultats. Cependant, la littérature nous a apporté des précisions sur les points clés utiles dans la reconnaissance d'image.

Figure 12 : Proposition de points clés pour la détection des mains



Source : Ortega (2019)

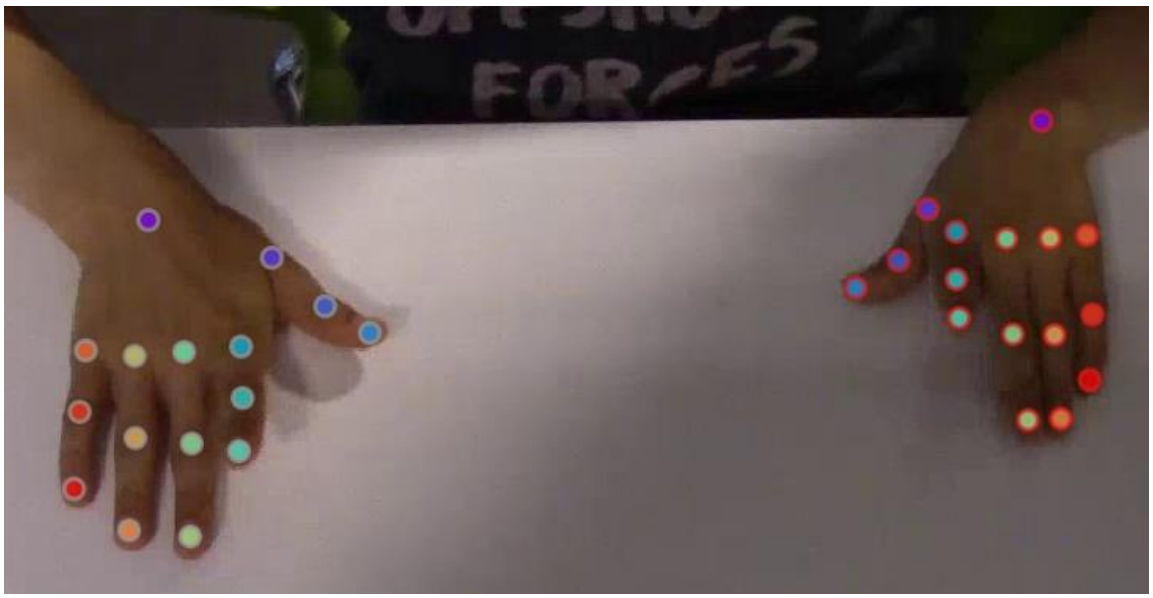
Par rapport à ce que nous avons fait, il faudrait rajouter un point entre chaque phalange intermédiaire et proximale et entre chaque phalange distale et intermédiaire (Figure 12). De plus, nous devrions aussi rajouter un point supplémentaire au niveau du centre du poignet (Ortega, 2019).

Les scientifiques ayant créé ou utilisé DeepLabCut affirment que plus il y a de points clés, plus le modèle sera performant (Lauer, et al., 2021, p. 9; Mathis, et al., 2018, p. 1283).

> *4 labels on a mouse is [sic] better than 2! 8 is [sic] better than 4! This is especially true with multiple animals that have occlusions. Have two mice that are on top of each other? Our networks learn who-is-who, and what point-belongs-to-who, but you need enough points! When in doubt, adding a few extra key points will only help - you can ignore them in your analysis later if you want* (DeepLabCut, 2021f).

Comme les points entre les phalanges distales et intermédiaires (exemples aux numéros 19 et 20, Figure 12) sont parfois trop rapprochés sur les vidéos en notre possession, nous avons fait le choix de rajouter uniquement les points entre les phalanges intermédiaires et proximales par rapport à notre prototype pour l'améliorer (Figure 13). Ceci empêchera les confusions ou chevauchements des points lors de la labélisation.

Figure 13 : Dernier prototype maDLC-2 avec labels



Source : Photo de l'auteur adaptée de Jaspers et al. (2016)

4.1.7. Amélioration du modèle

Après avoir créé un deuxième prototype avec les nouveaux points clés, les performances par rapport à son prédécesseur peuvent être évaluées (Tableau 2).

Tableau 2 : Évaluation des modèles maDLC

Modèle	Batch size	Training iterations:	% Training dataset	Shuffle number	Train error (px)	Test error (px)	p-cutoff used	Train error with p-cutoff	Test error with p-cutoff
maDLC-1	8	10’000	95	1	2,49	4,73	0,60	2,49	3,87
maDLC-1	8	20’000	95	1	1,94	3,44	0,60	1,94	3,44
maDLC-2	8	20’000	95	1	1,98	3,40	0,60	1,98	3,39

Source : Données de l’auteur

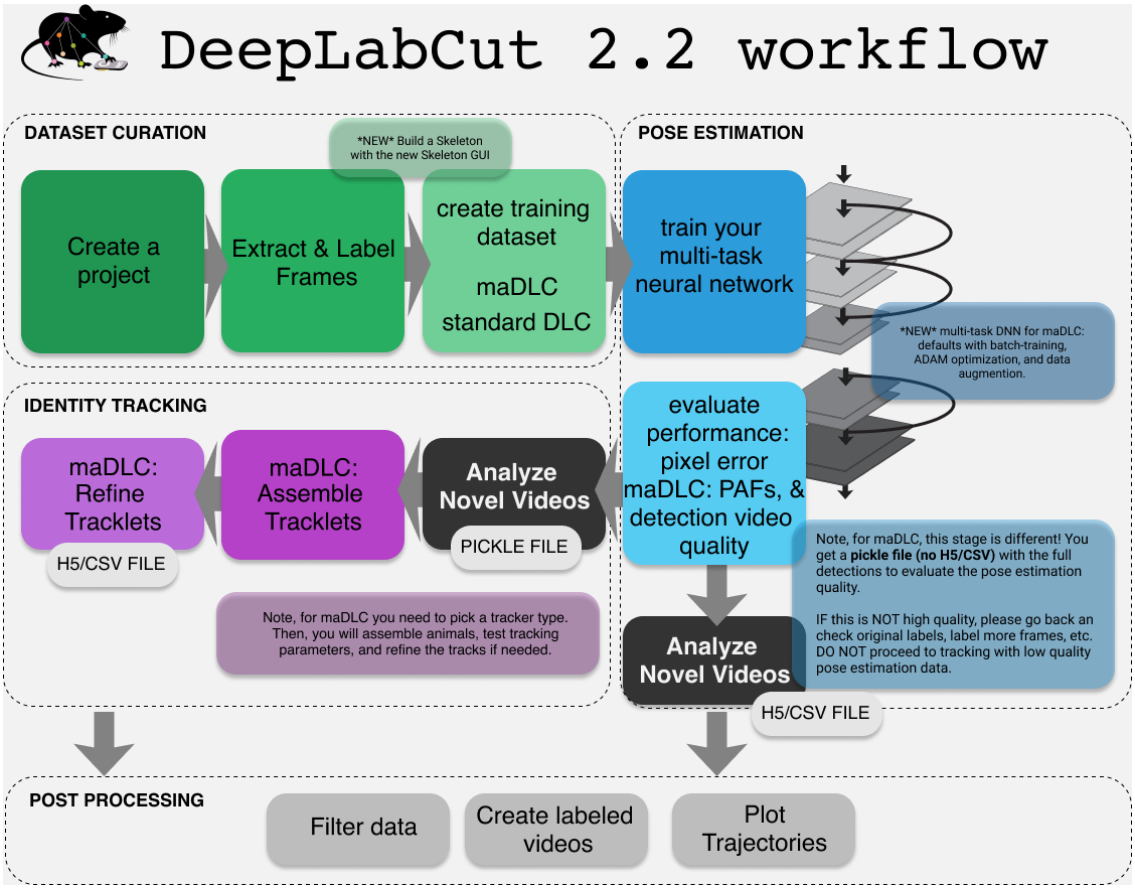
Une ligne a été rajoutée au début du tableau afin de montrer la progression de performance entre 10'000 et 20'000 itérations. Nous observons une nette diminution des erreurs en pixels. En revanche, alors que notre nouveau prototype aurait dû surpasser l’ancien, il est à peine plus performant. Le modèle le moins récent est plus compétitif avec les données d’entraînement. Malgré cela, ce qui nous intéresse davantage ici est le taux d’erreur pour les éléments de tests.

Certes, la différence semble minime entre les deux prototypes, mais notre choix se portera sur le plus récent. En effet, nous sommes convaincus qu’il sera plus pertinent, car il correspond aux règles et aux conseils de la littérature.

4.2. Modèle DeepLabCut

Les fondateurs et les éditeurs de DLC ont préparé plusieurs processus afin d’aider à obtenir le meilleur modèle possible. Ceux-ci décrivent chaque étape et prévoient même les échecs, ainsi que la manière de les résoudre au mieux ou de comprendre leurs causes.

Figure 14 : *Workflow* recommandé par DeepLabCut pour l'usage multi-animal



Source : Lauer et al. (2021)

Avant de commencer, il est important de contrôler s'il existe déjà un prototype capable de réaliser ce que nous souhaitons. Dans ce but, DeepLabCut met à disposition des modèles[17] déjà entraînés sur des animaux et des scénarios spécifiques. Ceux-ci peuvent être utilisés pour l'analyse vidéo sans avoir besoin de les entraîner nous-mêmes, car ils sont déjà prêts pour cette utilisation (DeepLabCut, 2021g).

Actuellement, il existe uniquement des modèles standards appelés aussi *single* animal. La seule disponibilité est une estimation de pose complète du corps humain, car il n'en existe pas pour une partie spécifique telle que les mains. Cette possibilité ne convient donc pas à notre étude, ce qui fait que nous devons créer notre propre modèle.

Dans un premier temps, nous utiliserons le GUI pour l'extraction, la labélisation des frames et la création du *training dataset*. Puis, nous ferons usage de Colaboratory afin d'entraîner et d'évaluer notre réseau de points. Cette première étape nous permettra d'obtenir un modèle

[17] Disponible sur http://www.mackenziemathislab.org/dlc-modelzoo

qui servira à l'analyse de nouvelles vidéos et pour en tirer des données utiles à la quantification des mouvements en miroir.

4.2.1. Création du projet

Pour créer un projet, nous lançons le GUI comme indiqué à la fin de l'installation pour Windows 10 (4.1.2). Les données obligatoires à l'élaboration d'un projet sont : son nom, l'expérimentateur et les vidéos à labéliser. Parmi les options supplémentaires, nous retrouvons la sélection de l'emplacement du projet, la copie des vidéos et le projet multi-animal. Nous avons tout coché, ce qui nous a permis de définir l'endroit de stockage, d'y inclure directement les vidéos dans le même lieu et d'avoir un type d'expérimentation qui peut contenir plusieurs animaux.

Avant de procéder à la prochaine étape, nous devons modifier la configuration pour qu'elle corresponde à ce que nous souhaitons effectuer. Avec n'importe quel éditeur de texte, nous devons ouvrir le fichier de paramétrisation « config.yaml » créé dans le projet.

Dans ce document, nous allons personnaliser les éléments utiles à la détection. Pour ce faire, nous définissons les individus. Nous aurons deux mains qui contiendront exactement les mêmes points clés. Pour toutes les variables écrites dans ce manuel de configuration, nous utiliserons la convention nommée *lower camel case*, c'est-à-dire qu'une variable commence par une minuscule et chaque nom rajouté commencera par une majuscule. Les *items* de paramétrisation par défaut de DeepLabCut ne seront pas touchés et seront en minuscules.

```
individuals:
- leftHand
- rightHand
```

Ensuite, nous déterminons nos points clés en tant que « *body parts* ». Ceux-ci ont été déterminés au point 4.1.6.

```
multianimalbodyparts:
- wristCenter
- thumbStart
- thumbCenter
- thumbEnd
- indexStart
- indexCenter
- indexEnd
- middleStart
- middleCenter
- middleEnd
- ringStart
- ringCenter
- ringEnd
- littleStart
- littleCenter
- littleEnd
```

Nous laisserons la configuration « *identity* » à la valeur *false*. Elle aurait pu être mise comme *true* si nous avions un signe distinctif sur l'une des mains et que nous souhaitions que le logiciel la reconnaisse grâce à cela. Un petit triangle aurait pu être dessiné sur les mains par exemple, mais cela n'est pas nécessaire et rajouterait une condition supplémentaire pour les futures vidéos analysées. Nous noterons que le paramètre *multianimalproject* est défini comme *true* puisque nous travaillons sur un projet maDLC.

```
    # Project definitions (do not edit)
Task: maDLC_MM_hands
scorer: Brice
date: Jul8
multianimalproject: true
identity: false
```

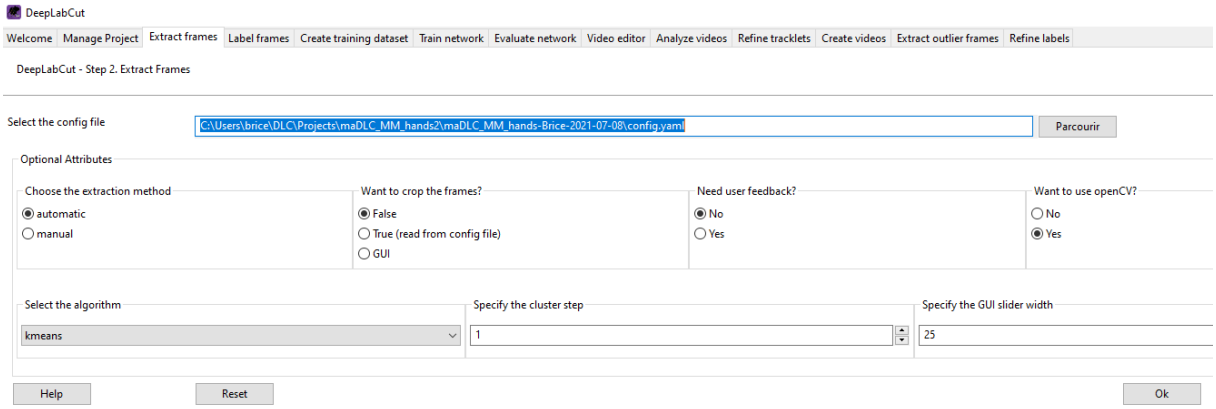
Il est possible de configurer le squelette directement dans le fichier, mais il est plus facile de le paramétrer plus tard dans l'interface graphique.

4.2.2. Extraction et labélisation des images

DLC va extraire 20 images pour chaque vidéo que nous avons choisie pour l'entraînement. Celles-ci seront ensuite labélisées pour donner les connaissances nécessaires d'apprentissage au programme.

Le GUI est réparti en plusieurs onglets pour procéder à une étape après l'autre. Nous allons nous intéresser à la partie « *Extract frames* » (Figure 15). Les options par défaut étant celles qui nous conviennent, il suffit alors d'appuyer sur « Ok » et laisser le logiciel travailler. Pour suivre la progression du programme en tout temps, il faut se rendre dans la fenêtre de terminal d'Anaconda.

Figure 15 : Aperçu de l'interface graphique de DLC

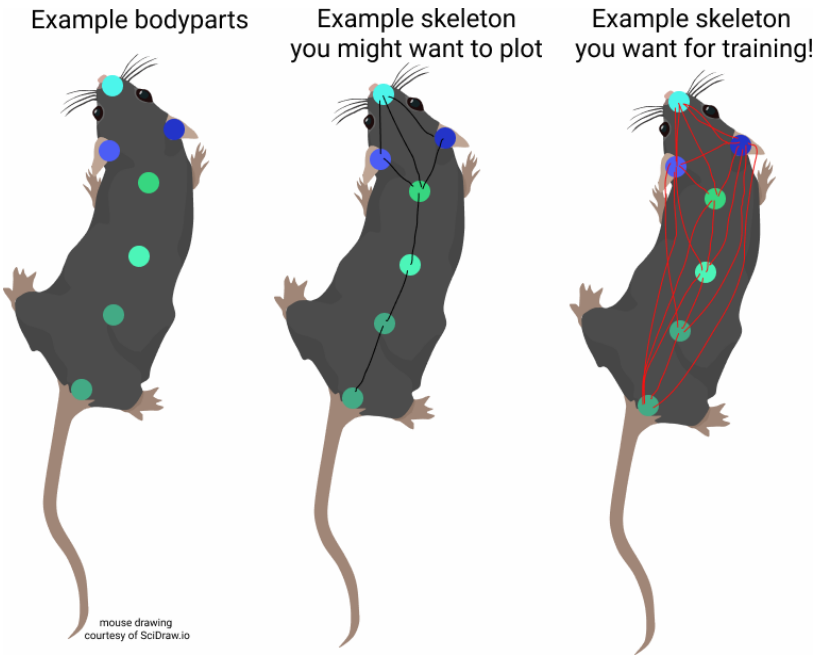


Source : Capture d'écran de l'auteur

Maintenant que les images sont extraites, nous pouvons les labéliser. L'onglet « *Label frames* » nous permet de lancer un nouveau GUI. Il ne nous reste plus qu'à placer les points clés à leur place sur les 20 images de chaque vidéo. C'est un travail relativement long et qui requiert une bonne concentration pour placer les labels aux bons endroits de manière systématique. Plus la précision est grande, plus l'algorithme sera performant. Une illustration de cette interface graphique est disponible (Annexe IV).

Nous pouvons désormais construire le squelette. Nous serions tentés d'en réaliser un de manière naïve, ce qui signifie relier certains points entre eux pour obtenir une représentation humaine (Figure 16).

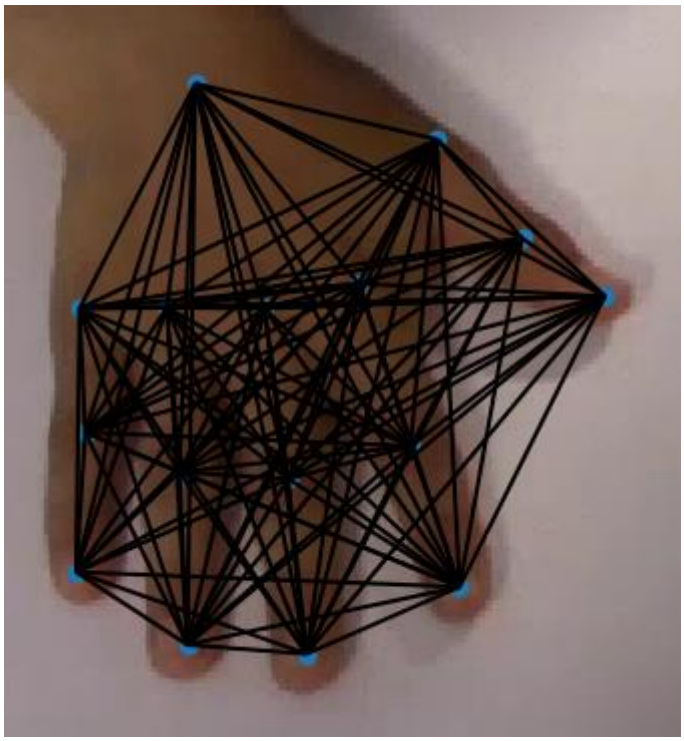
Figure 16 : Exemple de squelette pour entraînement



Source : DeepLabCut (2020)

Contrairement au schéma mental que nous avons, le but est de relier tous les points entre eux (Figure 16, Figure 17) (DeepLabCut, 2020a).

Figure 17 : Squelette hyperconnecté pour une main



Source : Photo de l'auteur

Cette option a été supprimée dans la version la plus récente soit la *2.2rc3*, car il n'était pas pertinent de faire relier par l'utilisateur quelque chose qui peut être réalisé automatiquement par le programme.

4.2.3. Création du jeu de données d'entraînement

Nous passons désormais sur Colaboratory, même si la création du jeu de données d'entraînement pourrait aussi se faire sur le GUI. Pour transférer les données, nous avons déposé notre projet maDLC-2 sur un *repository* GitHub[18]. Ensuite, nous avons créé une copie d'une démonstration[19] réalisée par DeepLabCut. Finalement, nous l'avons adaptée[20] pour correspondre à nos besoins.

Cette suite de commandes commence par installer DLC dans l'environnement d'exécution, pour ensuite télécharger (*git clone*) le répertoire en local. Des variables sont configurées pour les différents chemins nécessaires (vidéos, fichier de configuration).

[18] Disponible sur https://github.com/Bearbrice/maDLC_MM_hands2

[19] Disponible sur https://bit.ly/3hJcmM9

[20] Disponible sur https://bit.ly/3A7jng9

Il est maintenant possible de créer le *dataset* d'entraînement à l'aide de l'API DeepLabCut.

```
deeplabcut.create_multianimaltraining_dataset(path_config_file, windows2linux=True)
```

4.2.4. Entraînement du réseau

Le *dataset* étant prêt, le réseau peut maintenant être entraîné pour devenir un modèle. Pour cela, nous paramétrons la variable *shuffle* qui équivaut à un. Ce chiffre est défini par défaut lorsque rien n'est précisé quand le jeu de données est créé.

Le paramètre *displayiters* permet d'afficher la progression de l'entraînement dans la console. Le réglage *saveiters* sert à faire une sauvegarde de l'évolution en cas de déconnexion du GPU ou d'un autre problème. Finalement, il reste à définir quand nous souhaitons arrêter l'algorithme avec *maxiters* (DeepLabCut, 2021e).

```
shuffle = 1
deeplabcut.train_network(path_config_file, shuffle=shuffle, displayiters=250,
saveiters=2500, maxiters=20000)
```

4.2.5. Évaluation du réseau

Si la donnée *plotting* est activée, toutes les images de test et d'entraînement sont affichées avec les étiquettes mises manuellement et celles qui ont été prédites (DeepLabCut, 2021e). L'évaluation du réseau nous renverra un fichier plat qui pourra être converti et analysé comme aux points 4.1.5 et 4.1.7

```
deeplabcut.evaluate_network(path_config_file, Shuffles=[shuffle], plotting=True,
c_engine=False)
```

Nous pouvons extraire des indices (*scoremaps, locref layers, and part affinity fields*[21]) d'évaluations supplémentaires à l'aide de la commande suivante (DeepLabCut, 2020b).

```
deeplabcut.extract_save_all_maps(path_config_file, shuffle=shuffle, Indices=[0])
```

You need to cross validate parameters before inference. Here, you will run the new function (below) that will smartly try to optimize your inference_config.yaml file. You can also manually edit this file afterwards (more below). But, this first part will validate the parameters

[21] Souvent abrégé en « PAFs ».

and optimize either hits/misses, RMSE, and percent correct keypoints (tracking we deal with below) (DeepLabCut, 2020b).

```
deeplabcut.evaluate_multianimal_crossvalidate(
        path_config_file,
        Shuffles=[shuffle],
        edgewisecondition=True,
        leastbpts=1,
        init_points=20,
        n_iter=100,
        target='rpck_train',
    )
```

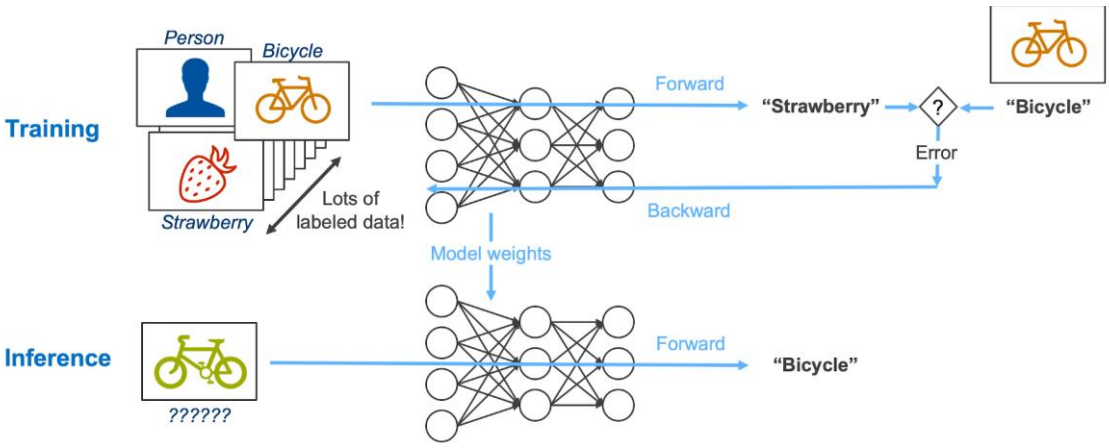
À la fin de cette étape, nous disposons d'un modèle entraîné et capable d'être utilisé pour l'analyse vidéo et pour la création d'images avec suivi du mouvement.

4.3. Utilisation du modèle pour l'analyse vidéo

Afin d'estimer la pose avec DeepLabCut à partir de l'apprentissage effectué, nous devons premièrement effectuer une évaluation sur une vidéo définie (*input*). Dans un deuxième temps et en se basant sur les résultats (*output*), il est possible de reprendre l'élément donné en entrée pour y rajouter les labels. La dernière phase consiste à extraire les trajectoires pour en faire des graphiques puis créer une vidéo qui affiche les labels.

Les prochaines étapes symbolisent le processus d'inférence. Pour rappel, il s'agit de la capacité qu'a le logiciel à prédire les points clés à partir des enseignements dont il dispose.

Figure 18 : Schéma de relation entre l'entraînement et l'inférence



Source : Intel (2021)

Pour réaliser cette partie, nous exportons le modèle entraîné[22] dans le *cloud* Google Drive. Le fait de séparer l'étape d'entraînement de celle de l'inférence est utile pour disposer de procédures distinctes, mais aussi pour avoir une sauvegarde du modèle que nous avons élaboré.

Nous travaillerons à nouveau avec Colab pour l'environnement d'exécution, mais cette fois avec un nouveau *notebook*[23]. Il est nécessaire d'installer DLC, récupérer le répertoire depuis le serveur distant et configurer les variables contenant les différents chemins pour disposer des outils de travail adéquats.

4.3.1. Analyse d'une vidéo

Les vidéos destinées à être évaluées ne doivent pas être confondues dans le fichier de configuration avec celles qui étaient prévues pour l'entraînement. Pour ce faire, il suffit de les placer dans un dossier au choix et de définir une variable comprenant le chemin pour y accéder.

L'algorithme d'analyse vidéo peut maintenant être démarré. Les résultats sont stockés dans des fichiers « *pickle*[24] » localisés dans le même répertoire que celui où se trouve la vidéo (DeepLabCut, 2021h).

```
videofile_path =
['/content/drive/MyDrive/ColabFiles/maDLC_MM_hands2/Inference_videos']

scorername = deeplabcut.analyze_videos(path_config_file,
                                       videofile_path,
                                       shuffle=shuffle,
                                       videotype=VideoType,
                                       c_engine=False)
```

Pour vérifier que les détections brutes fonctionnent correctement, nous pouvons déjà extraire une première vidéo.

```
Specific_videofile =
'/content/drive/MyDrive/ColabFiles/maDLC_MM_hands2/Inference_videos/VPers1.mp4
'
deeplabcut.create_video_with_all_detections(path_config_file, [Specific_videofile],
scorername)
```

[22] Disponible sur https://drive.google.com/file/d/16CD9S9LwTHifLPJ0dTAecmLRR7eDNpnB/view?usp=sharing

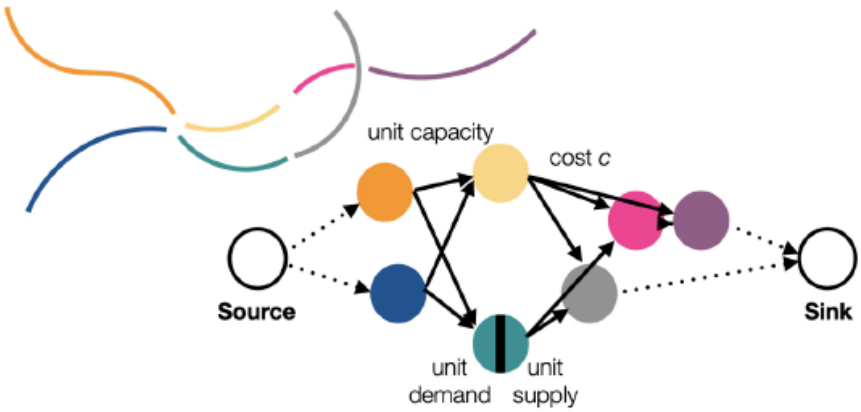
[23] Disponible sur https://bit.ly/2VppD3v

[24] Extension de fichier pour des données issues d'un processus par lequel une hiérarchie d'objets Python est convertie en un flux d'octets (*byte stream*) (Python Software Foundation, 2021).

L'analyse a produit des détections dans un fichier sérialisé en Python. Nous devons maintenant les assembler et effectuer le suivi des mains. Plusieurs types de *trackers* sont disponibles (*box*, *skeleton* ou *ellipse*) et il faut les tester pour savoir lequel utiliser (DeepLabCut, 2021e). Comme il s'agit pour l'instant de la compréhension de l'analyse, *tracking type* nommé *box* est choisi. Plus tard, il sera envisageable de déléguer ce choix à l'utilisateur tout en émettant une recommandation.

La méthode se nomme « *detections2tracklets* », car elle convertit ce qui a été détecté en courtes *tracks* (pistes) de cinq ou six images généralement (deadcode, 2019). Selon la définition les *tracklets* (Figure 19) sont représentés « *as nodes of a graph, whose edges encode the likelihood that the connected pair of tracklet belongs to the same track* » (Lauer, et al., 2021, p. 6).

Figure 19 : Représentation de nœuds symbolisant les *tracklets*



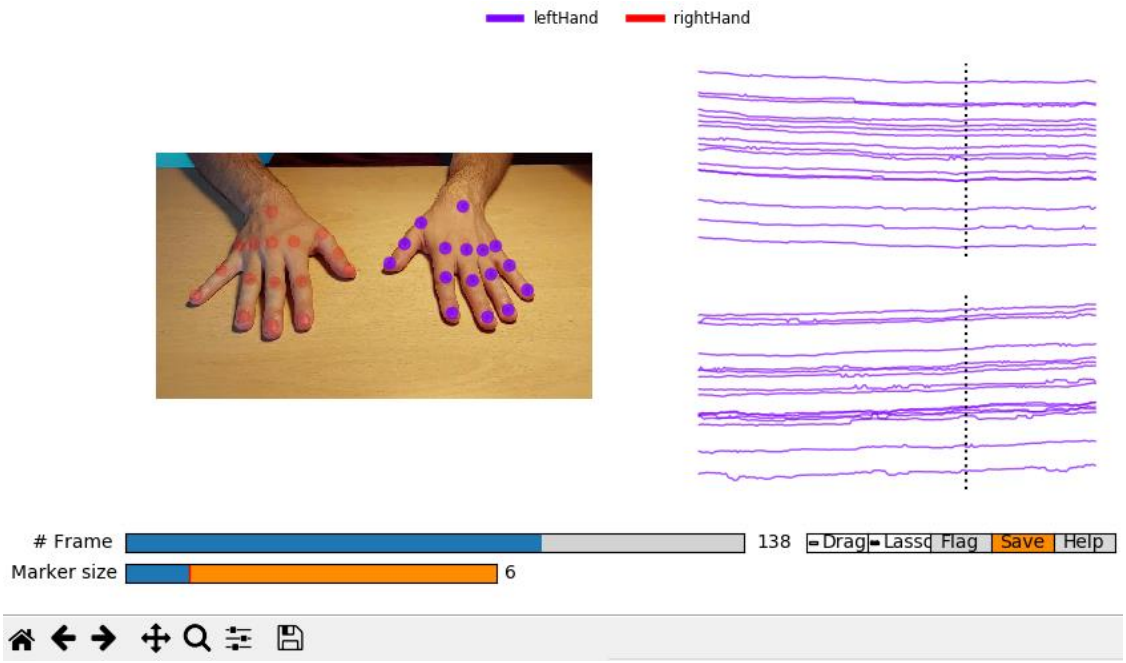
Source : Lauer et al. (2021)

```
tracktype= 'box' #box, skeleton, ellipse

deeplabcut.convert_detections2tracklets(path_config_file, Specific_videofile,
videotype=VideoType, shuffle=shuffle, track_method=tracktype, overwrite=True)
```

Une fois la commande précédente réalisée, nous pouvons analyser dans le GUI de DeepLabCut si les labels ont été placés correctement et, si nécessaire, les corriger (Figure 20).

Figure 20 : GUI pour affiner les *tracklets*



Source : Capture d'écran de l'auteur

Pour convertir les données d'analyse en « *tracks* », nous devons les extraire à l'aide de la commande « *convert_raw_tracks_to_h5* ». Cela nous permettra d'obtenir un fichier avec le suivi des mouvements.

```
picklefile='/content/drive/MyDrive/ColabFiles/maDLC_MM_hands2/Inference_videos/V
Pers1DLC_resnet50_maDLC_MM_handsJul8shuffle1_20000_bx.pickle'

vid='/content/drive/MyDrive/ColabFiles/maDLC_MM_hands2/Inference_videos/VPers1.
mp4'

deeplabcut.convert_raw_tracks_to_h5(path_config_file, picklefile)
```

Les données brutes peuvent maintenant être filtrées et un fichier plat que nous utiliserons pour la quantification sera créé.

```
deeplabcut.filterpredictions(path_config_file,
                             videofile_path,
                             videotype=VideoType,
                             track_method = tracktype)
```

Les trajectoires peuvent être tracées avec la ligne d'exécution « *plot_trajectories* ». Cette étape nous permet de créer des représentations graphiques de l'analyse vidéo. « *The plotting components of this toolbox utilizes matplotlib. Therefore, these plots can easily be customized by the end user. We also provide a function to plot the trajectory of the extracted poses across the analyzed video* » (DeepLabCut, 2021e).

```
deeplabcut.plot_trajectories(path_config_file,videofile_path, videotype=VideoType,
track_method=tracktype)
```

Lorsque la commande sera effectuée, un dossier « *plot-poses* » est disponible. En fonction des résultats obtenus, nous pouvons modifier le « *p-cutoff* » pour n'avoir que les points que le logiciel trouve avec une probabilité élevée.

4.3.2. Création d'une vidéo labélisée

Il est possible de créer une vidéo au format « .mp4 » avec des étiquettes prédites par le modèle, ce qui permet de voir si nos différents points clés ont été détectés et suivis correctement. Cette vidéo est enregistrée dans le même répertoire que celui où se trouve la vidéo originale (DeepLabCut, 2021e).

```
deeplabcut.create_labeled_video(path_config_file,
                                videofile_path,
                                shuffle=shuffle,
                                draw_skeleton=True,
                                videotype=VideoType,
                                save_frames=False,
                                filtered=True,
                                track_method = tracktype)
```

4.4. Quantification

4.4.1. Données extraites

L'analyse vidéo de l'estimation de pose renvoie un jeu de données en format texte ouvert. L'extension du fichier de sortie est en *comma-separated values* (CSV) et les données sont délimitées par des points-virgules. Pour visualiser les résultats d'une manière plus organisée et, si souhaitée, graphique, il existe des outils, notamment Microsoft Excel, qui sont capables de les afficher sous forme de tableaux ou de graphiques.

Les informations sont représentées en trois colonnes par point suivi (exemple colonnes B, C et D, Figure 21). Deux colonnes indiquent les coordonnées sur l'image exprimées par les axes X et Y. Par exemple, le pouce gauche se trouve à 337,50 pixels sur l'axe X et à 275,50 pixels sur l'axe des Y. La dernière colonne affiche le taux de confiance (*likelihood*), c'est-à-dire la probabilité que l'algorithme exprime pour s'assurer qu'il s'agisse bien du point recherché. Celui-ci est compris entre 0,01 qui indique peu de probabilités qu'il s'agisse du point clé jusqu'à 1,00 qui représente une certitude que ce soit bien celui-ci.

Chaque ligne du document représente une image (*frame*) de la vidéo et donc un axe temporel. Lorsque DLC ne trouve pas un point, ce qui équivaudrait théoriquement à un *likelihood* de zéro, il laisse vide la ligne de données qui concerne le point clé[25] non détecté (Figure 21).

Figure 21 : Exemple de données

	A	B	C	D	E	F	G	H	I	J
1	scorer	DLC_resnet5	DLC_resnet5	DLC_resnet5	DLC_resnet5	DLC_resnet5	DLC_resnet5	DLC_resnet5	DLC_resnet5	DLC_resnet5
2	individuals	leftHand	leftHand	leftHand	leftHand	leftHand	leftHand	leftHand	leftHand	leftHand
3	bodyparts	wristCenter	wristCenter	wristCenter	thumbStart	thumbStart	thumbStart	thumbCente	thumbCente	thumbCente
4	coords	x	y	likelihood	x	y	likelihood	x	y	likelihood
296	291	556.0	92.9375	1.0	487.0	137.125	1.0	461.25	188.125	0.864
297	292	556.0	92.9375	1.0	487.0	137.125	1.0	461.25	188.125	0.8267
298	293	556.0	92.9375	1.0	487.25	137.125	1.0	461.25	188.125	0.92
299	294	556.0	94.0625	1.0	487.75	136.25	1.0			
300	295	556.0	94.0625	1.0	487.75	136.25	1.0			
301	296	556.0	94.0625	1.0	487.25	137.0	1.0			
302	297	556.0	94.0625	1.0	486.5	137.0	1.0			
303	298	554.0	94.0625	1.0	481.0	146.75	0.9995			
304	299	554.0	94.6875	1.0	477.0	147.75	0.9985			
305	300	554.0	95.0625	1.0	476.75	147.75	0.97			
306	301	554.0	95.0625	1.0	476.5	147.25	0.97			
307	302	554.5	95.0625	1.0	476.5	147.25	0.989			
308	303	554.5	94.625	1.0	476.5	147.25	0.9917			
309	304	554.5	94.0625	1.0	476.5	147.25	0.9585			
310	305	554.0	94.0625	1.0	476.5	147.25	0.847			
311	306	554.0	93.9375	1.0	476.5	148.0	0.847			

Source : Photo de l'auteur

4.4.2. Situation

Dans le cas où nous comparerions le bout du pouce gauche à celui du droit, il est important de comprendre que ce sont deux points différents et nous nous retrouverions donc avec six colonnes à traiter. Dans un premier temps, nous devons comprendre les complexités liées à ce système à trois variables (X, Y et temps). Il nous est facile de comparer les axes X et Y à l'aide d'un graphique en nuages de points sans tenir compte du temps.

Nous pouvons alors traiter séparément les variables et les comparer (X et temps ou Y et temps), mais les mettre en relation semble difficilement concevable. Avec une belle amplitude verticale dans la vidéo, une extraction intéressante des données serait possible. En comparant le bout du pouce gauche au droit par exemple, des graphiques mesureraient le mouvement.

4.4.3. Solution

Nous avions déjà choisi dans le rapport (point 4.1.3) un exercice pour une raison particulière. En effet, ce dernier dispose de mouvements d'une grande ampleur sur l'axe des Y et quasiment aucun relatif à l'axe X. Nous sommes donc en mesure de quantifier uniquement

[25] Soit les données X, Y et le likelihood.

ce qui nous intéresse, soit les mouvements à la verticale tout en laissant de côté les mesures horizontales.

Les données récoltées (exemple Figure 21) doivent être exploitées comme dans un processus *Extract Load Transform* (ELT), qui est une variante de l'architecture de système d'information connue nommée *Extract Transform Load* (ETL). Les informations seront d'abord extraites par DeepLabCut, puis chargées dans notre *workflow* automatisé et finalement transformées avant d'être disponibles pour l'utilisateur.

4.4.4. Mise en œuvre

L'utilisation de Python dans Colab pour l'analyse de vidéo permet de continuer avec ce langage de programmation pour le développement de la quantification. En effet, celui-ci est très utilisé dans la communauté scientifique notamment pour le traitement des données. L'un des outils les plus populaires mais aussi l'un des plus puissants se nomme « pandas ».

« *pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language* » (pandas, 2021).

Les *packages* « numpy » et « matplotlib » seront aussi installés pour la visualisation et la modification des informations. Avant d'appeler directement les commandes développées par ces utilitaires, il faut les installer et les importer comme dans l'exemple suivant. Afin de simplifier leur appel, nous utiliserons le code « *as* » suivi du nom, dans le but de leur créer un raccourci.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

L'extraction des données brutes a déjà été réalisée par DeepLabCut dans un fichier plat comme expliqué précédemment. Pour le chargement des informations en mémoire, il suffit d'utiliser la commande en Python ci-dessous.

```
pd.read_csv('Extracted_Data.csv')
```

Une fois les données chargées, la transformation peut commencer. Pour ce faire, il faut réaliser les étapes suivantes :

1. Remplacer les noms de colonne par des nombres ;
2. Choisir les colonnes à conserver ;
3. Supprimer les lignes inutiles en en-tête ;

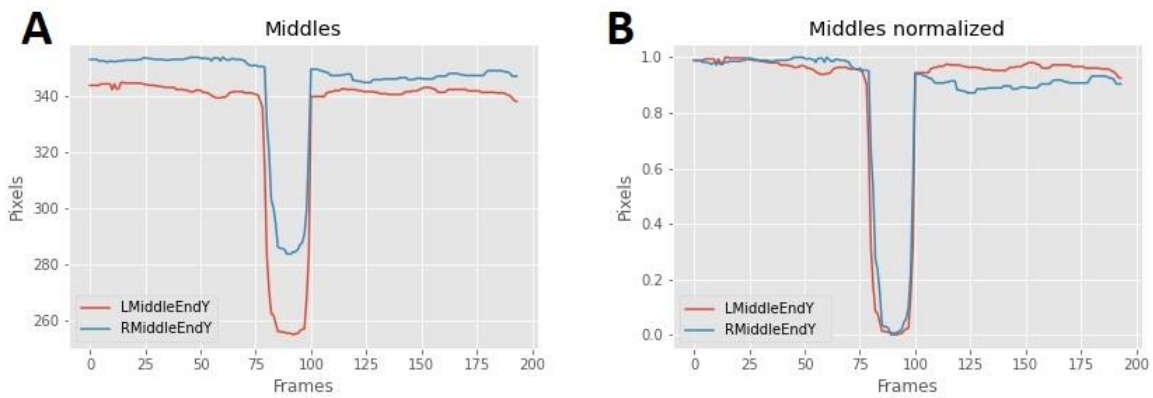
4. Renommage des colonnes avec des noms.

L'étape deux va sélectionner les données que nous souhaitons conserver. Dans le cas de la quantification des mouvements en miroir, les points avec l'amplitude la plus importante sont à comparer. Dans l'exercice choisi (point 4.1.3), ce sont les bouts des doigts qui auront ces caractéristiques précises.

Après les quatre étapes citées, un *dataframe* nettoyé est prêt pour le sectionnement de l'information. Il consiste à séparer en plusieurs trames de données ce qui est existant pour ensuite séparer et effectuer un traitement sur les différents points à garder, soit les phalanges distales pour chaque main. Nous allons donc créer de nouveaux *dataframes* qui compareront, par exemple, le bout du pouce de la main gauche (*LThumbEndY*) à celui de la main droite (*RThumbEndY*). De plus, nous conserverons la colonne « *coords* » qui représente les frames et la notion de temps dans la vidéo. Les phases suivantes sont réalisées dans une méthode en Python :

1. Séparation des données ;
2. Conversion en *float*[26] ;
3. Suppression des lignes vides (Figure 21) ;
4. Ajout de colonnes normalisées ;
5. Création de graphiques pour chaque comparaison standard (Figure 22A) et normalisée (Figure 22B).

Figure 22 : Exemple de graphiques en sortie



Source : Photo de l'auteur

[26] Obligatoire pour réaliser le point suivant.

La normalisation a été choisie (Figure 22B), car elle permet la remise à l'échelle des valeurs dans une plage comprise entre zéro et un. Il est important de normaliser les courbes pour qu'elles soient comparables avec des alignements similaires.

Figure 23 : Formule de normalisation

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Source : Schéma de l'auteur

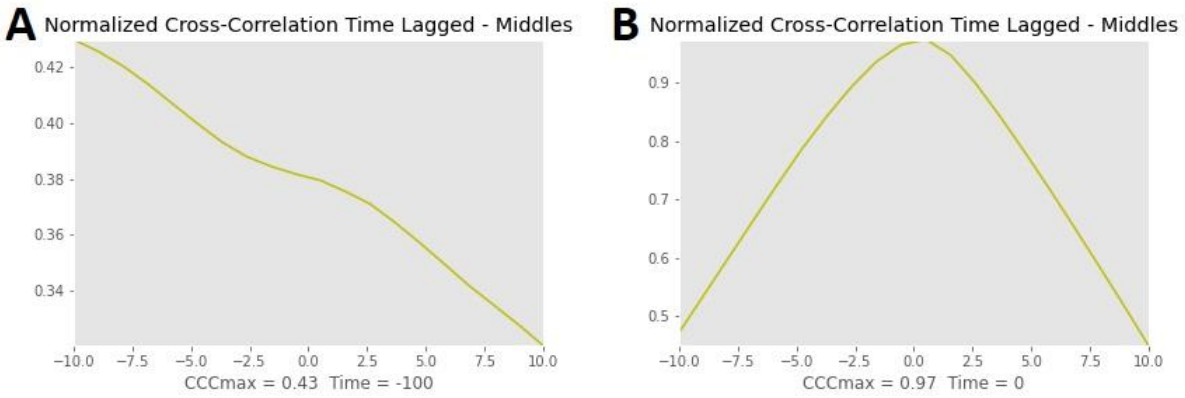
À l'aide de la fonction permettant la normalisation (Figure 23), une méthode a été créée en Python pour la réaliser facilement.

```
#Function to normalize a column
def normalize(col):
  min = col.min()
  max = col.max()
  return (col - min) / (max – min)
```

Afin de montrer si nos courbes sont similaires, ce qui diagnostiquerait des mouvements en miroir, nous allons utiliser le principe de base de la corrélation. Pour rappel, le but est d'indiquer à l'utilisateur le niveau de MM mesuré. Dans notre cas, nous avons des séries qui évoluent dans le temps (*time series data*). Il est donc possible d'utiliser le *Pearson correlation coefficient*. Celui-ci mesure la covariation de deux signaux continus dans le temps et indique la relation linéaire sous la forme d'un chiffre compris entre moins un pour une corrélation négative, zéro pour une non-corrélation et un pour une corrélation parfaite (Cheong, 2020). Ce chiffre nous donne donc une mesure pour estimer les mouvements en miroir.

Cependant, cette technique renvoie uniquement à un chiffre sans aucune représentation visuelle. Pour corriger cela, l'utilisation de la *cross-correlation* peut être intéressante. Tout d'abord, on peut utiliser une *Time lagged cross-correlation* (TLCC) qui permet d'identifier la directionnalité entre deux signaux, par exemple « *a leader-follower relationship in which the leader initiates a response which is repeated by the follower* » (Cheong, 2020).

Figure 24 : Comparaison de la corrélation croisée décalée dans le temps



Source : Photo de l'auteur

Les graphiques (Figure 24) comparent les extrémités des majeurs des mains selon deux simulations effectuées. Des courbes presque linéaires se retrouvent chez les personnes n'ayant pas de mouvement en miroir (Figure 24A). Au contraire, une courbe montante puis descendante avec un pic maximal aux alentours du temps zéro (Figure 24B) se dessine pour les personnes atteintes de MM pour une hypothèse de quatre, soit le maximum selon l'échelle de Woods et Teuber. L'inscription « CCCmax » sous les graphiques correspond au *cross-correlation coefficient maximum*.

Finalement, tous les graphiques seront rendus accessibles pour les personnes qui utiliseront l'interface graphique et la moyenne du coefficient de corrélation de Pearson sera calculée pour chaque doigt. Selon les essais réalisés, les moyennes correspondraient au niveau suivant sur l'échelle de Woods et Teuber :

- 0 : < 0,50 ;
- 1 : >= 0,50 ;
- 2 : >= 0,66 ;
- 3 : >= 0,82 ;
- 4 : >= 0,96.

Ces scores ont été déterminés sur la base de deux essais en condition réelle, avec une personne non affectée simulant des mouvements en miroir, puis en réalisant l'exercice sans MM. Une fois les vidéos analysées, la moyenne des *Pearson correlations coefficients* a été calculée. De ce fait, à partir du maximum et du minimum récoltés, des paliers ont été calculés pour établir les cinq niveaux. La corrélation et ses différents dérivés sont des formules qui permettent de comparer deux signaux et ont déjà été utilisées lors de différentes études sur les mouvements en miroir (Klingels, et al., 2015, p. 737; Zielinksi, et al., 2018, p. 1549; Magne, et al., 2021, p. 737).

Pour que la quantification et la détection se déroulent dans les meilleures conditions possibles, une documentation expliquant les enjeux du cadrage de la vidéo ainsi que la position des mains à adopter a été rédigée (Annexe VIII). Elle pourra ensuite être réutilisée dans l'application web pour donner des indications claires à l'utilisateur.

4.5. Application web

Afin de réaliser l'application web, une architecture devra être choisie pour définir où l'exécution à l'aide d'un GPU aura lieu, soit en interne, soit de façon délocalisée. Après avoir déterminé celle-ci, la configuration du serveur sera effectuée, suivie du développement de l'application web. Finalement, le déploiement sera mis en œuvre et expliqué.

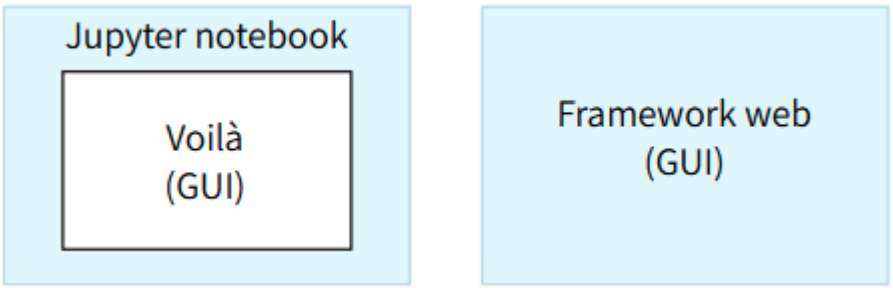
4.5.1. Architectures

Comme expliqué précédemment (3.2.2), nous avons le choix entre un serveur physique ou virtuel pour réaliser une application interactive sur internet. Ce travail se focalisera sur un environnement dématérialisé. Les possibilités d'architecture se concentreront, pour le backend, à l'appel d'une API pour l'exécution sur un GPU se trouvant dans le *cloud* ou par l'hébergement de l'ensemble de l'application dans une seule et même machine virtuelle.

Les concepts qui seront développés dans ce travail peuvent aussi être repris plus tard sur un serveur physique avec ou sans unité de traitement graphique. Le fait d'avoir une infrastructure propriétaire permet une stabilisation des coûts et à long terme leur amortissement. Effectivement, faire appel à des solutions *cloud* engendre des frais qui peuvent être plus ou moins importants en fonction des fournisseurs. Certains proposent des offres tout-en-un, alors que d'autres séparent la structure des dépenses en plusieurs catégories comme le stockage, la mémoire vive appelée RAM, les CPU ou encore les GPU[27]. Les montants à investir peuvent être fixes pour une quantité, comme un emplacement de stockage ou, le plus souvent, au *prorata* de l'utilisation d'un service par l'utilisateur. Un tableau des prix appliqués à ce jour par une liste non exhaustive de distributeurs de services *cloud* a été réalisé (Annexe V).

[27] Exemple disponible sur https://puzl.ee/cloud-kubernetes/computing-resources

Figure 25 : Interfaces graphiques proposées



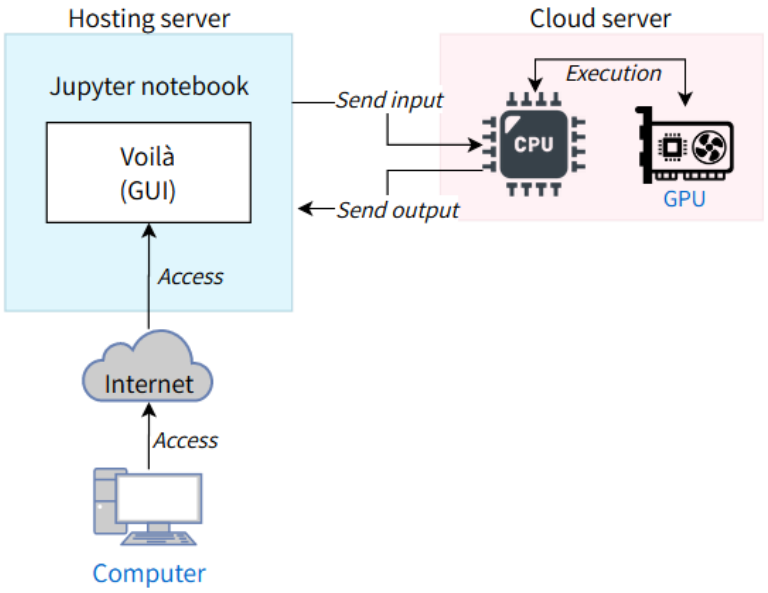
Source : Schéma de l'auteur

Avant toute chose, il est préférable de définir le *frontend* à utiliser (Figure 25). Python est le langage de programmation choisi, puisque DeepLabCut est basé sur ce dernier et qu'il a aussi été utilisé lorsque la quantification a été développée. Concernant le *framework* à utiliser, il en existe de nombreux, mais une solution légère sera préférable. Effectivement, Django, pour le plus connu, offre de multiples possibilités comme l'authentification ou la navigation, mais est trop lourd pour ce que nous souhaitons réaliser. Dans notre cas d'utilisation et pour avoir une application web efficace, une seule page bien structurée avec toutes les fonctionnalités nécessaires suffit. D'autres *frontends* plus légers sont disponibles, mais nécessitent de revoir tout ce qui a été fait jusqu'à présent pour gérer l'intégration. Le souhait est d'afficher de la manière la plus graphique possible les résultats. Pour parvenir à un tel résultat, il existe des moyens de développement appelés les *data dashboard,* qui reprennent le principe d'un écran de contrôle où tout doit être visuel. Conçus pour les analyses, l'interaction ainsi que pour la représentation de données en Python, les *frameworks* Dash, Streamlit, Shiny et Voilà se partagent le marché (Schmitt, 2020).

La solution la plus avantageuse et efficiente reste l'utilisation des développements existants, à savoir ce qui a été réalisé dans les *notebooks* Jupyter. L'opportunité pour accomplir ceci est d'utiliser Voilà[28], un moteur graphique qui emploie des widgets interactifs.

[28] Disponible sur https://github.com/voila-dashboards/voila

Figure 26 : Architecture de communication avec un serveur virtuel



Source : Schéma de l'auteur

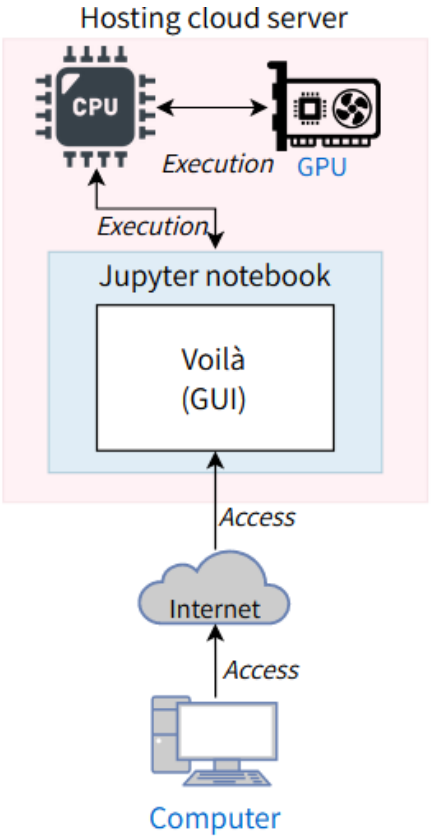
La première architecture présentée (Figure 26) représente le transfert d'une vidéo depuis le *frontend* vers un *backend* qui se trouve dans le *cloud*. Le fichier est acquis puis traité par un serveur d'exécution virtuel possédant un GPU. Les résultats des analyses et la vidéo labélisée sont ensuite renvoyés vers l'hôte. Cette exécution décentralisée permet d'utiliser des solutions de multiples fournisseurs tels que Google Cloud, Azure (Microsoft), AWS (Amazon), DigitalOcean et bien d'autres. Diverses entreprises apparaissent aussi sur le marché en proposant des offres diversifiées ainsi que des *packages* avec des solutions déjà installées et prêtes à être utilisées.

Pour arriver à cette fin, un nouvel entrant sur le marché du *machine learning* se nomme Gradient[29] et met en avant des services innovants. Conçu par l'entreprise Paperspace, il propose de nombreuses options comme les *notebooks* exécutables dans le *cloud*, mais aussi de déployer un modèle et d'y accéder via une API (Paperspace, 2021). Cette fonctionnalité est particulièrement intéressante, puisqu'elle allie des connaissances en *data science* et des infrastructures performantes. Malheureusement, celle-ci est encore en phase de développement puisque les « *Workflows and Deployments are currently in private beta. We'll be releasing them to the general public in the next 60 days* » (B. Lamson, employé chez Paperspace, communication personnelle, 21 juillet 2021). Cette solution ne peut donc pas être envisagée, pour l'instant, sous sa forme de déploiement d'API.

[29] Disponible sur https://gradient.paperspace.com/

Pour notre prototype, le principal avantage d'une infrastructure basée sur la délocalisation de l'exécution sur GPU est son coût. En effet, elle permet de démarrer la machine qui procédera à l'inférence à distance et de l'éteindre une fois le processus terminé. Les frais liés à l'utilisation d'une unité de traitement graphique sont généralement calculés au temps d'utilisation, ce qui permet des économies. En revanche, une perte de temps qui peut être considérable est à noter lors de ce processus, car il faut compter le temps de démarrage du serveur. Les étapes sont longues et nécessitent, dans le cas d'une mise en production, une attention particulière sur la sécurité lors des transferts des données entre les deux acteurs. Les communications ont été conceptualisées pour un éventuel développement dans un exemple (Annexe VII).

Figure 27 : Architecture pour un hébergement sur un serveur *cloud* avec GPU



Source : Schéma de l'auteur

La deuxième architecture (Figure 27) représente un hébergement de notre application web sur un serveur virtuel possédant une unité de traitement graphique qui permet de centraliser et rassembler le *frontend* et le *backend*. Il existe plusieurs *providers* pour ce service dont Saturn Cloud, Genesis Cloud, Puz.lee et Gradient. Ce type d'architecture permet de supprimer la perte de temps occasionnée dans le cas des communications avec un serveur externe puisqu'il est déjà allumé et prêt à l'exécution durant tout le processus. Ce dernier argument

est aussi son plus grand défaut, car la ressource de traitement graphique reste en fonction et engendre donc des coûts auprès du fournisseur.

Des essais rapides ont été réalisés avec les deux architectures auprès des distributeurs de services suivants :

- Première architecture : Genesis Cloud, Puzl.ee, Azure, Saturn Cloud, Paperspace ;
- Deuxième architecture : Google Cloud, Puz.lee, Genesis Cloud, Gradient, Saturn Cloud.

Les résultats de ces tests confirment nos hypothèses puisque l'architecture sous forme de communication avec un serveur externe n'est pas ergonomique à cause de l'attente causée par le démarrage des machines à distance. De plus, cela implique la mise en œuvre d'un service sécurisé de transfert de données. Il pourrait se faire soit grâce à un protocole d'accès à distance *Secure Shell* (SSH) à la machine contenant le GPU soit grâce à un serveur de gestion de fichiers via le protocole SSH File Transfer Protocol (SFTP). Une alternative serait d'utiliser un serveur d'intégration[30] qui déclencherait un processus d'exécution à chaque fois qu'il recevrait un nouveau fichier sur son lieu de stockage SFTP. Dans tous les cas, des moyens techniques supplémentaires doivent être déployés pour l'implémentation de cette infrastructure.

La deuxième architecture a, elle aussi, confirmé les recherches effectuées en amont, puisque les distributeurs de services *cloud* proposent des infrastructures Jupyter notebook avec GPU prêtes à l'utilisation. Cela rend donc le travail beaucoup plus efficace et orienté vers le but de notre application web. De ce fait, ce moyen est choisi pour la réalisation de notre interface graphique. Pour effectuer un choix technique quant au fournisseur à utiliser, il est souhaitable de disposer d'un accès au serveur grâce à une adresse Internet Protocol (IP) publique, et non uniquement à l'interface de développement Jupyter. En effet, cela permet de gérer les installations des logiciels complémentaires comme DeepLabCut et Voilà directement sur le système d'exploitation et non pas dans des *notebooks* Jupyter. De plus, une condition importante est que l'hébergement de la machine soit compatible avec le règlement général sur la protection des données (RGPD). À cette fin, notre décision s'est orientée vers Genesis Cloud[31] qui offre cette possibilité avec un *data center* à Reykjavik, en Islande[32].

[30] Par exemple Microsoft BizTalk Server.

[31] Plus d'informations sur https://www.genesiscloud.com/

[32] Ce pays ne fait pas partie de l'union européenne mais applique la RGPD (EFTA, 2018).

4.5.2. Configuration du serveur

Pour commencer, une machine virtuelle, nommée « instance » par le fournisseur choisi, est créée sur la console d'administration des services de notre compte sur Genesis Cloud. Lors de la conception de cette dernière, trois choix de processeur graphique NVIDIA de la série GeForce sont proposés : GTX 1080Ti, RTX 3080 et RTX 3090. La première option convient à nos besoins techniques en termes de mémoire GPU (3.1.6) et est la plus abordable avec un prix de 0,60 dollar (0,55 CHF[33]) par heure. La case qui permet de mettre les pilotes du GPU sur l'instance doit être cochée.

Une autre possibilité est le choix parmi le catalogue de paquets préinstallés. D'abord, il faut choisir le logiciel (Docker, Jupyter Notebook ou JupyterLab). Pour ce travail, la décision se porte sur JupyterLab qui est une version plus complète du *notebook* Jupyter classique. Après cette étape, des images de logiciels supplémentaires sont disponibles, dont TensorFlow (TF), librairie *open-source* de *deep learning* développé par Google et utilisé dans DeepLabCut. L'emploi de la version *1.15* de TF sera préféré puisqu'elle est la mieux supportée (DeepLabCut, 2021i). L'instance est désormais prête et, après avoir confirmé la création, il faut attendre environ cinq minutes pour que les pilotes de la carte graphique s'installent (Sprung, 2020). Le système d'exploitation est Ubuntu *18.04.05 LTS* par défaut et la version de Python est la *3.7.9*.

Une fois le serveur prêt, nous pouvons y accéder grâce au protocole SSH dans un terminal de commandes sur notre machine physique. Alors que JupyterLab est déjà configuré, l'extension Voilà ne l'est pas. Le gestionnaire de package « pip » peut l'installer facilement. Auparavant, d'autres prérequis sont nécessaires (Ng, 2021). Normalement, ils devraient être d'ores et déjà installés, mais, par principe de précaution, ils seront aussi ajoutés comme présenté ci-dessous.

```
ssh ubuntu@147.189.XXX.XXX
pip install widgetsnbextension
pip install ipywidgets
pip install voila
jupyter nbextension enable --py widgetsnbextension --sys-prefix
jupyter serverextension enable voila --sys-prefix
```

DeepLabCut peut maintenant être installé. Un problème[34] de package est connu lors de cette opération. En effet, dans une installation avec le *package manager* pip, il y a des

[33] Taux de conversion de 0.9146 sur https://www.telexoo.com/, le 28 juillet 2021 à 10 heures et 44 minutes.

[34] Ce problème est recensé sur https://github.com/DeepLabCut/DeepLabCut/issues/498

dependencies, soit d'autres éléments liés au bon fonctionnement du programme qu'il est obligé implicitement d'ajouter. L'une de ces extensions, « ruamel.yaml », n'a pas la bonne version et sa mise à jour doit être forcée. Il faut également baisser la version de « matplotlib » à *3.2.2* pour éviter d'avoir des avertissements dans la console de JupyterLab.

```
pip install deeplabcut
pip install ruamel.yaml --no-deps --ignore-installed
pip install matplotlib==3.2.2
```

À la suite de cette étape, le serveur est presque prêt, mais il émet encore une erreur à propos de la *CUDA Deep Neural Network library* (cuDNN) : « *Failed to get convolution algorithm. This is probably because cuDNN failed to initialize, so try looking to see if a warning log message was printed above* » (Project Jupyter, logiciel JupyterLab, 2021).

La source de cette erreur est la version installée *7.3.1* de *cuDNN* dans l'environnement de base Anaconda se trouvant sur le serveur Ubuntu (Bahramudin, 2019). Il faut donc la désinstaller et lancer la commande inverse, soit l'installation pour obtenir la version *7.6.5.32* qui est la plus récente sur la distribution proposée par ce gestionnaire de *packages* lors de la réalisation de ce travail.

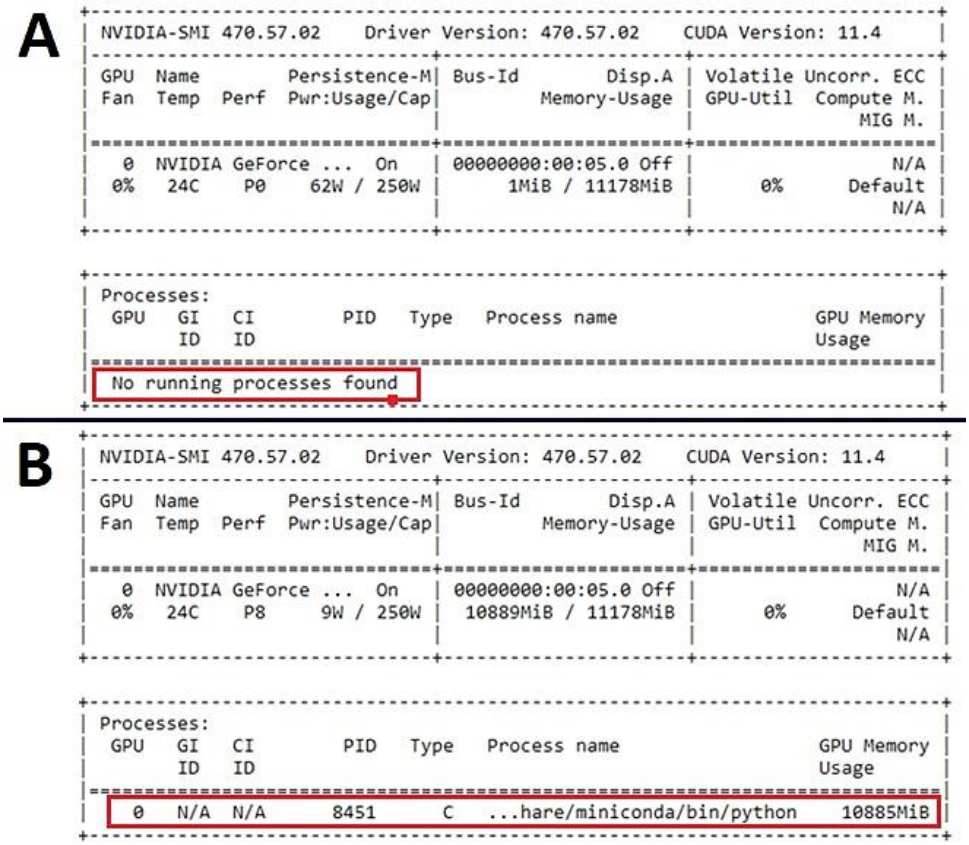
```
conda uninstall cudnn
conda install -c conda-forge cudnn
```

Figure 28 : Détails fournis sur la console Genesis Cloud

Source : Capture d'écran de l'auteur

Tout est maintenant réglé et prêt à être utilisé. Pour accéder à JupyterLab, la console de Genesis Cloud génère une adresse internet spécifique (Figure 28) sur un proxy qui permet de se connecter avec n'importe quelle machine. La commande « nvidia-smi » sert à obtenir des informations sur le GPU NVIDIA (Figure 29).

Figure 29 : Opération nvidia-smi dans l'invite de commande



Source : Captures d'écran de l'auteur

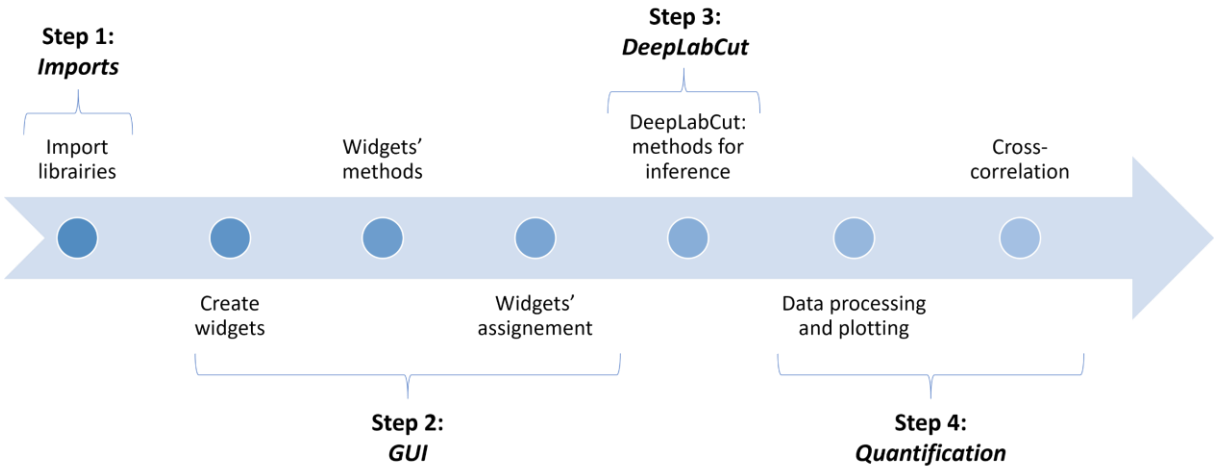
La carte graphique peut maintenant être testée dans l'onglet qui s'ouvre automatiquement, afin de vérifier qu'elle fonctionne avec DeepLabCut. Il faut d'abord contrôler qu'aucun autre processus n'utilise le GPU (Figure 29A), avant de faire un essai. Après, DLC peut être importé puis une analyse vidéo lancée. Celle-ci s'exécute dans le navigateur, indépendamment du terminal. De ce fait, pendant son déroulement, il est possible de relancer la commande dans notre console de commandes système et d'observer le résultat (Figure 29B).

4.5.3. Développement

Le code de notre application web reprendra les différentes parties développées jusqu'à maintenant, soit l'inférence et la quantification. En plus de ces moyens, l'interface graphique doit être simple et compréhensible. Les fonctions à développer font partie intégrante du *product backlog* (Annexe VI).

Le concept de l'application à réaliser permet à un utilisateur de télécharger vers le serveur une vidéo et d'exécuter une analyse complète avec le suivi du mouvement et la quantification. Les résultats sont ensuite affichés et sont disponibles pour le téléchargement.

Figure 30 : Structure du code sur le *notebook* Jupyter



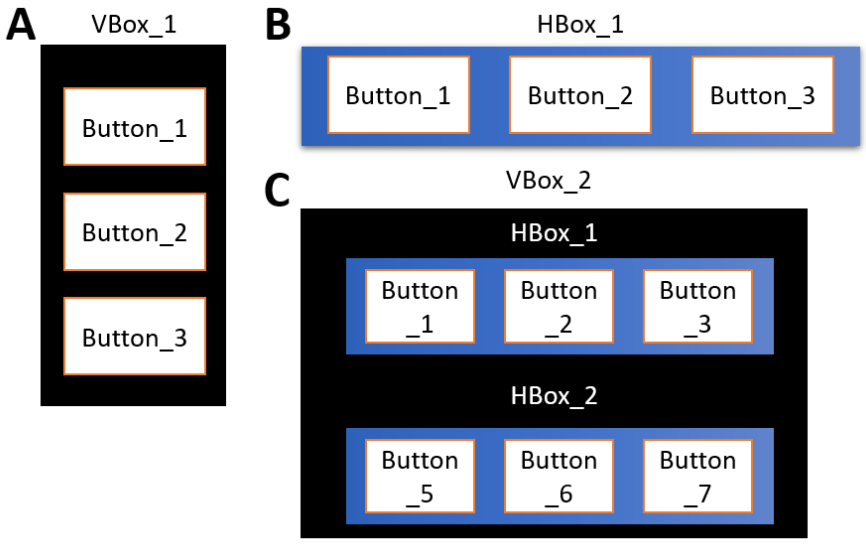
Source : Schéma de l'auteur

Le code est structuré en sept parties (Figure 30) qui sont représentées graphiquement grâce aux *widgets* de « ipywidgets[35] ». Ceux-ci sont présents dans la console de sortie de Jupyter en temps normal, mais grâce au moteur graphique Voilà, ils seront affichés sur notre application web sans code apparent.

Premièrement, il est nécessaire d'importer toutes les librairies utiles de DeepLabCut à ipywidgets en passant par « numba » qui permet la gestion des ressources du GPU et quelques autres outils. Pour rappel, aucune installation supplémentaire n'est requise, comme le serveur a déjà été configuré au point précédent (4.5.2).

[35] Documentation disponible sur : https://ipywidgets.readthedocs.io/en/stable/

Figure 31 : Mise en page ipywidgets



Source : Schéma de l'auteur

Deuxièmement, la partie graphique est programmée avec la création des *widgets* et leurs méthodes. Il en existe de nombreux, tels que des boutons, des listes déroulantes ou encore des conteneurs graphiques d'images. Après les avoir instanciés, leur positionnement sur la page peut être défini : il s'agit du *layout* de l'application. Dans ce but, la librairie graphique *ipywidgets* propose des *Vertical Boxes* (VBox) et des *Horizontal Boxes* (HBox) (Project Jupyter, 2020). Ces mises en pages (Figure 31) constituent le sens dans lequel les éléments seront disposés soit de manière verticale (Figure 31A), soit horizontale (Figure 31B). Il est aussi possible d'imbriquer les « boxes » (Figure 31C) ce qui permet de mieux gérer la disposition globale de multiples éléments.

Troisièmement, le code réalisé pour l'inférence dans Colaboratory (4.3) est repris et adapté dans des méthodes. Lorsque les analyses de DeepLabCut seront lancées, le serveur attribuera automatiquement de la mémoire de l'unité de traitement graphique. En revanche, il ne la libérera pas et, pour cette raison, les commandes suivantes doivent être exécutées à cette fin.

```
cuda.select_device(0)
cuda.close()
```

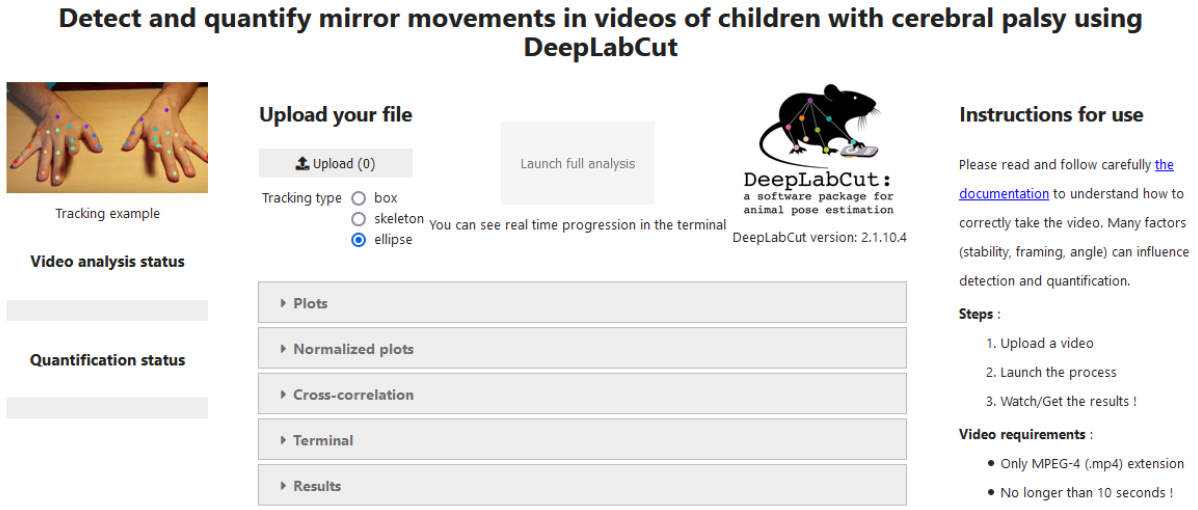
Finalement, le processus de quantification est lui aussi repris (4.4.4) et adapté. Sa mission supplémentaire est d'afficher, après le traitement des données, les résultats de façon représentative pour l'utilisateur. Afin de mettre en place cette tâche, les graphiques extraits tout au long de l'évaluation des mouvements en miroir sont sauvegardés provisoirement sur

le serveur sous forme d'images. Ils seront ensuite affectés à un *widget* pour être visualisés dans l'application web en toute fin de procédure.

4.5.4. Démonstration

L'application web a été conçue en trois parties graphiques (Figure 32) sous forme de boxes verticales : le suivi de la progression du processus, le panneau central constitué de la boîte à outils avec les *outputs* et le dernier composant qui indique les instructions d'utilisation du programme.

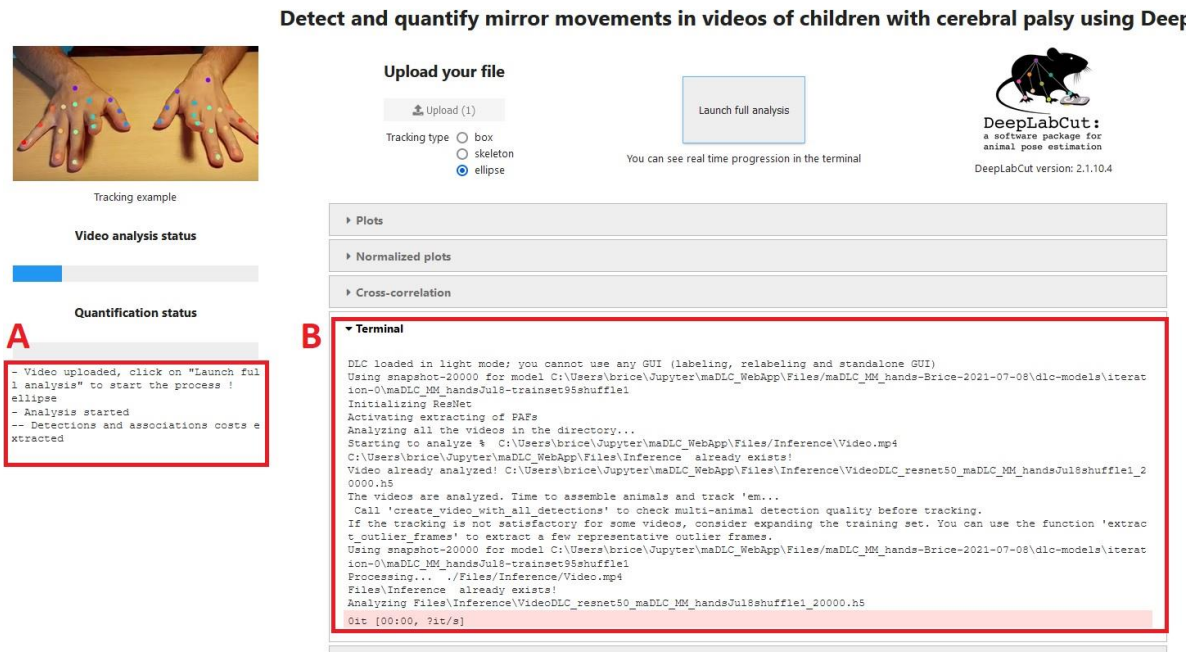
Figure 32 : Aperçu de l'application web



Source : Application web de l'auteur

Un *widget* pour le téléchargement vers le serveur (*upload*) est disponible pour l'utilisateur (Figure 32). Au clic, un explorateur de fichier natif de l'OS concerné s'ouvre, afin de pouvoir naviguer vers le fichier désiré. Seules les vidéos au standard MPEG-4, représenté par l'extension « .mp4 », sont autorisées, car il s'agit du type de codage le plus utilisé et courant pour la capture d'images, notamment sur la plupart des smartphones. Lorsque le serveur a bien reçu le fichier, le programme désactive le *widget* d'*upload* et active le bouton « *Launch full analysis* » pour le rendre cliquable et donc pour lancer l'analyse. Avant cela, l'utilisateur a le choix entre les trois types de *tracking type* proposé par DeepLabCut. Par défaut, c'est l'option ellipse qui est choisie, puisqu'elle est recommandée par les créateurs du logiciel (DeepLabCut, 2021e).

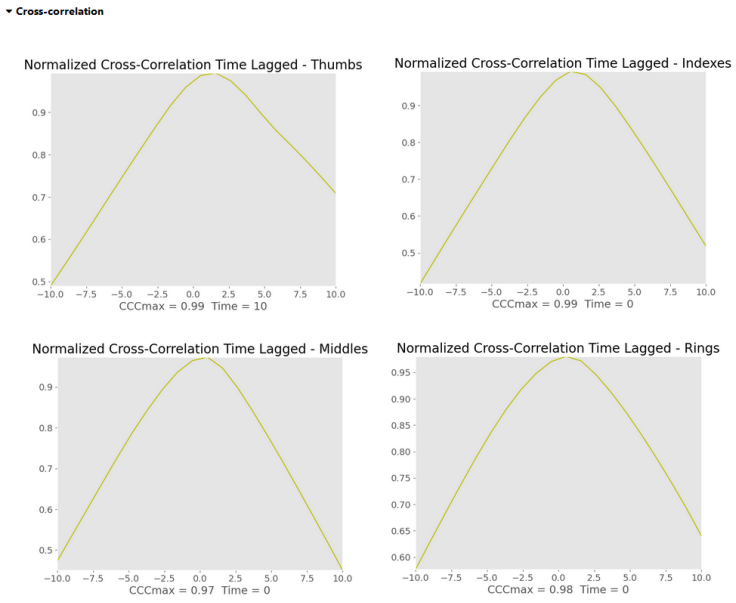
Figure 33 : Emplacement des consoles de progression



Source : Application web de l'auteur

Lorsque le processus est lancé avec le bouton cité auparavant, l'analyse vidéo est déclenchée et deux fenêtres d'affichage de la progression sont disponibles : l'une pour voir les étapes réalisées (Figure 33A) et l'autre pour voir en temps réel ce qui est effectué dans le terminal (Figure 33B) c'est-à-dire l'*output* original des méthodes Python comme dans une console système.

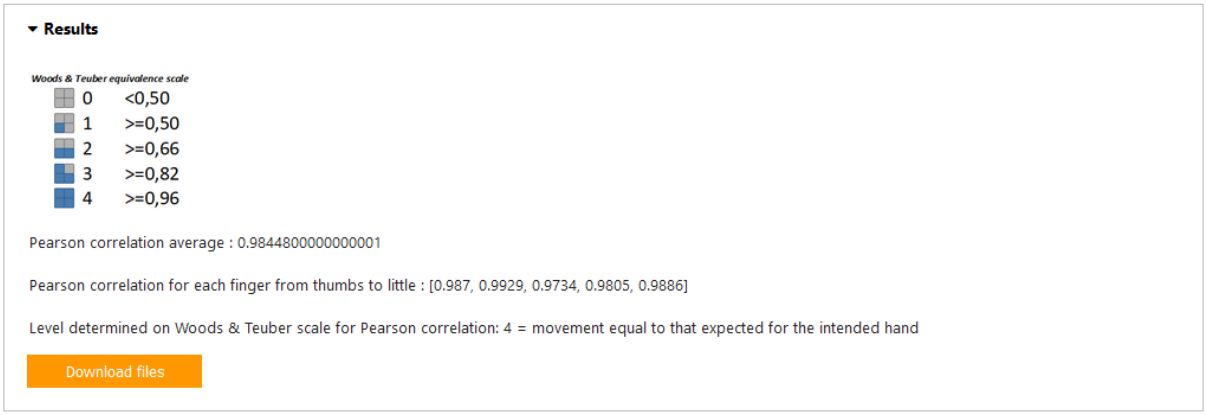
Figure 34 : Aperçu des graphiques de cross-correlation



Application web de l'auteur

Des barres graphiques de progression montrent également l'évolution en temps réel de l'analyse vidéo et de la quantification. À la fin des tâches, les résultats sont disponibles dans les accordéons (exemple Figure 34), qui sont des outils graphiques ouvrants et fermants. Par exemple, l'un d'eux propose de voir les graphiques de *cross-correlation*.

Figure 35 : Fenêtre de résultats



Application web de l'auteur

Un accordéon dédié aux résultats et à la mesure sur l'échelle de Woods et Teuber (1.4) est disponible (Figure 35). Il indique le niveau de mouvements en miroir perçu lors du processus de quantification. Comme les résultats ne sont pas stockés durablement, l'utilisateur peut les télécharger afin de les conserver sur une machine locale. Le fichier obtenu sous le format ZIP contient les graphiques, la vidéo de sortie ainsi que les données brutes de DeepLabCut sous format CSV.

4.5.5. Déploiement

Genesis Cloud génère un lien où est lancé le serveur JupyterLab. L'extension Voilà nous permet de modifier ce lien en rajoutant le suffixe « /render/voila/ » suivi du nom de fichier du *notebook*. L'application web peut alors être lancée sans accéder au code. Les connexions simultanées se font sur le même serveur et les sessions ne sont pas gérées pour être coordonnées. C'est pourquoi, dans sa version expérimentale, cette application ne peut accueillir qu'une personne à la fois. De plus, pour une exécution simultanée avec plusieurs utilisateurs du processeur graphique et, par définition dans notre cas, sa mémoire, il faudrait développer un système de mise en attente. Ainsi, si la ressource n'est pas disponible, la personne souhaitant utiliser le service en est informée.

5. ÉVALUATION DES RÉSULTATS

L'application web a joué un rôle clé puisqu'elle a permis d'assembler toutes les pièces de ce travail. Ce qui a été réalisé peut maintenant être évalué. Dans un premier temps, l'aptitude de détection va être mise à l'épreuve. Ensuite, la quantification sera analysée sur des cas réels afin d'en constater sa pertinence et ses limites. Finalement, la technologie développée pour l'application web va être inspectée pour comprendre les intérêts qui en découlent et les enjeux d'une mise en production.

5.1. Capacité de détection

DeepLabCut est un logiciel d'estimation de pose qui s'est révélé précis et accompagné d'une documentation étoffée et d'une grande communauté. Il est régulièrement mis à jour, ce qui représente l'un de ses meilleurs atouts, mais aussi un inconvénient majeur quant aux versions documentaires qui ont de la peine à suivre le rythme. Supporté sur tous les OS, ce programme pose malgré tout des problèmes de compatibilité sur les systèmes Linux et en général. Les environnements virtuels comme Anaconda permettent heureusement de stabiliser ces inconvénients, même si cela nécessite parfois un peu d'altérations logicielles.

Dans les différentes observations effectuées sur des vidéos labélisées par le modèle entraîné dans ce travail, les points sont correctement suivis. En effet, le plus important était qu'il n'y ait pas de confusions entre les points comme le bout du majeur qui se retrouverait à la place de celui de l'annulaire. Il n'y a donc pas de faux suivi du mouvement sur de mauvais points. Lors de l'analyse de la vidéo, si le point devait être occulté parce que DeepLabCut était dans l'impossibilité de prédire celui-ci, alors il préférera le faire disparaître, plutôt que d'essayer de reconnaître un autre endroit qui ne serait pas le bon. Cela a son importance pour le processus de quantification afin d'éviter que la courbe de l'axe Y se retrouve faussée.

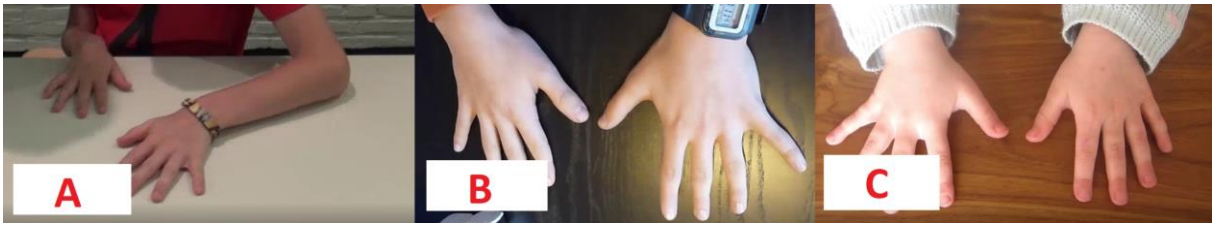
La décision quant au point à placer respecte le but à atteindre, en accord avec la littérature en termes d'estimation de pose qui recommande un certain nombre de points (4.1.6). Elle n'est cependant pas idéale dans la mesure où il aurait fallu rajouter un point supplémentaire sur chaque doigt ce qui aurait été difficile à réaliser avec les vidéos à notre disposition.

L'entraînement du modèle a été effectué avec 20'000 itérations. Pour améliorer ceci, nous pourrions augmenter le nombre d'itérations, mais surtout diversifier les vidéos. Ces dernières sont toutes issues du même exercice malgré des situations de capture changeantes. À cette fin, il serait souhaitable de se focaliser sur des images contenant des occultations de nos points. Cela permettrait de renforcer les capacités de prédictions du modèle.

5.2. Évaluation de la quantification

Dans le but d'évaluer notre processus de quantification, trois vidéos ont été sélectionnées (Figure 36). Cela permettra aussi de mesurer la qualité de la détection puisque ce sont des vidéos qui n'ont pas été utilisées pour l'entraînement du modèle.

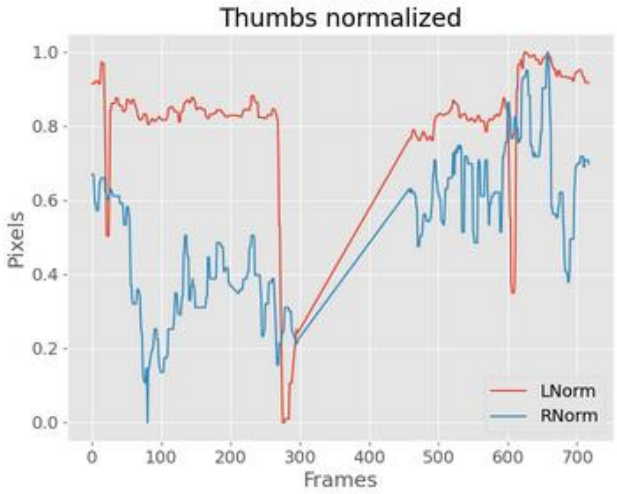
Figure 36 : Aperçu des vidéos et de leur cadrage



Source : Photos de l'auteur

La première vidéo (Figure 36A) est estimée de manière qualitative à trois sur l'échelle de Woods et Teuber (1.4). Les résultats obtenus montrent un coefficient de corrélation de 0,78 soit, selon nos mesures d'équivalence proposée (4.4.4), à un score de trois également. La seconde vidéo (Figure 36B) a été quantifiée à une mesure de zéro tout comme l'estimation qualitative. La troisième et dernière vidéo (Figure 36C) a quant à elle obtenu un score de zéro avec notre processus alors qu'elle aurait dû probablement obtenir la mesure égale à un sur l'échelle de référence. La quantification proposée est donc pertinente pour deux cas sur trois. Afin de découvrir pourquoi la vidéo quantifiée en dernier est incorrecte dans son score, il faut s'intéresser aux résultats graphiques.

Figure 37 : Évolution sur l'axe Y des pouces de la troisième vidéo évaluée



Source : Graphique de l'auteur

Le graphique (Figure 37) montre un saut important aux alentours de 270 *frames*. Pour expliquer ceci, il faut regarder la vidéo labélisée obtenue par l'analyse. Dans cette dernière, il

est possible de constater que le label du pouce gauche n'est pas découvert pendant un certain temps. Pourtant, les lignes ont été supprimées pour ne pas obtenir ce genre de résultat (4.4.4). L'explication est simple et a été vérifiée grâce à la vidéo : le programme a en fait perdu le point à un moment très haut sur l'axe Y et lorsqu'il l'a retrouvé il s'y trouvait très bas d'où ce bond impressionnant. Le problème est donc en rapport avec la détection et non la quantification qui est tributaire de cette situation. Il serait donc possible d'apporter de légères améliorations sur l'estimation de pose comme dit précédemment (5.1).

Pour conclure cette évaluation de la quantification, nous relevons que les résultats obtenus sont pertinents et très satisfaisants au vu des vidéos utilisées qui présentent des conditions de cadrage vidéo que nous n'avons pas pu choisir. La corrélation semble donc une méthode quantitative adéquate pour la mesure des mouvements en miroir.

Afin de garder une vue d'ensemble sur les possibilités de quantification, d'autres méthodes pourraient être envisageables. En effet, après discussion avec Madame H. Haberfehlner (Dr et associée de recherche à l'Amsterdam *University Medical Centers*, communication personnelle, 30 juin 2021), une possibilité aurait pu être de conserver uniquement les vidéos où les mains s'ouvraient et se fermaient. Effectivement, des occultations avaient lieu et il n'était donc visuellement plus possible de savoir où se situaient les points. Une hypothèse a alors été émise : si une main se ferme et l'autre aussi, alors le logiciel ne sera plus capable d'obtenir les points clés de la main censée être au repos, ce qui signifierait la présence de mouvements en miroir. Cette technique est en revanche plus difficile à échelonner, mais cela aurait été possible en calculant le temps d'occultation mais aussi son taux : si chaque doigt disparaît ou uniquement certains. Cependant, lors de cette même discussion, la technique de quantification préférée reste celle proposée dans ce rapport. L'utilisation de la corrélation a également été approuvée comme mesure pertinente.

5.3. Développement web

La création d'une application interactive sur internet a été considérée comme la meilleure façon de rendre accessible le plus aisément possible ce moyen aux cliniciens (3.2.1). Le défi était d'accélérer le processus d'inférence afin d'avoir un programme capable d'effectuer rapidement une analyse et d'afficher les résultats. Pour ceci, il a fallu intégrer une unité de traitement graphique. Cette tâche n'est pas aisée quand on ne dispose pas d'une infrastructure physique malgré les solutions *cloud* existantes. Cependant, avec un serveur avec un GPU, l'application développée a tenu ses promesses d'efficacité.

En revanche, malgré une satisfaction quant à l'exécution avec une unité de traitement graphique, c'est le fournisseur de service, Genesis Cloud, qui nous a surpris par sa façon de procéder. Effectivement, aucune garantie de disponibilité du service n'est fournie et lorsque toutes les cartes graphiques sont occupées dans le *data center* il n'est tout simplement plus possible de démarrer son instance pour une durée indéterminée. Aucun abonnement *premium* qui offrirait un accès en tout temps n'est à ce jour disponible auprès de cette entreprise.

Une variante avec CPU uniquement que nous avions initialement prévue semble complexe à développer et il faut tenir compte du temps d'exécution qui sera très lent. Le code utilisé sur le processeur graphique aurait pu être repris et adapté en changeant les chemins d'accès aux dossiers et fichiers et en supprimant les lignes qui concernent CUDA.

De plus, il aurait fallu des modifications importantes puisque le *kernel* Jupyter plante automatiquement lorsque la RAM est surchargée lors du développement en local avec une indication d'erreur expliquant : « *Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2* ». Nous émettons deux hypothèses : soit il s'agit d'un problème lié au CPU soit directement lié à Jupyter. Pour la première, Tensorflow doit être compatible avec les unités de centrale de traitement. Effectivement, chaque CPU utilise un type de jeu d'instructions particulier qui est *Advanced Vector Extensions* (AVX) pour les processeurs récents des marques Intel et AMD (Evanson, 2020). L'installation de composants et logiciels supplémentaires donc Bazel et Microsoft Visual C++ est obligatoire et doit être effectuée directement sur la machine (Venkatesh, Xu, & Tsai, 2021; Bazel, 2021). Pour la seconde hypothèse, nous estimons que le problème pourrait être directement lié à Jupyter Lab puisque lorsque la RAM est surchargée le noyau d'exécution plante automatiquement. Ce problème est déjà connu (jakes, 2019), mais nous avons cependant trouvé une solution de contournement. En effet, il serait possible d'utiliser MyBinder pour effectuer un déploiement dans un conteneur informatique et cela semble fonctionner. L'inconvénient majeur est donc que les tests en temps réels ne sont pas possibles puisqu'il faut à chaque fois envoyer les modifications au serveur et les compiler (*build*).

CONCLUSION ET RÉFLEXION

DeepLabCut est un logiciel d'estimation de pose intéressant et efficace pour suivre des êtres vivants ou des objets. Un modèle de détection et de suivi du mouvement a pu être entraîné sur un exercice précis défini dans cette étude. Grâce à ce dernier, il est possible de filmer les mains des enfants atteints de paralysie cérébrale unilatérale afin de quantifier si des mouvements en miroir sont présents et avec quelle importance. Effectivement, à la suite du processus d'inférence, des données brutes sur les coordonnées des différents points clés durant la vidéo analysée sont disponibles. Elles sont ensuite traitées avec des lignes de code en Python pour les transformer, les normaliser puis effectuer un coefficient de corrélation ainsi que la *cross-correlation*. Afin de rendre disponible l'ensemble de ce qui a été élaboré jusque-là, nous avons décidé de nous orienter vers une application web. Celle-ci est hébergée dans un serveur contenant une unité de traitement graphique pour utiliser la mémoire de cet élément. Elle améliore considérablement le temps d'analyse effectué avec les algorithmes de DLC. Les cliniciens sont donc en mesure de charger une vidéo vers le serveur, de lancer l'analyse et la quantification et d'obtenir des indicateurs aidant au diagnostic. Ce travail est expérimental et ne vise pas à remplacer, à ce stade, les observations médicales qualitatives, mais de les compléter avec une analyse quantitative afin de rendre l'évaluation la plus objective possible.

L'usage de DeepLabCut est un choix intéressant pour la création d'un nouveau modèle. Une autre possibilité aurait été de se tourner vers des exemples existants d'estimation de pose avec d'autres logiciels. Une analyse serait alors effectuée pour voir si la reconnaissance des points clés correspond au cas d'utilisation souhaité. Cette solution aurait permis de se focaliser sur la quantification sans se soucier de la détection et du suivi du mouvement déjà programmés.

Le processus d'évaluation des mouvements en miroir à l'aide des données brutes est pertinent, mais a malgré tout ses défauts. Effectivement, une attention particulière doit être portée aux conditions d'utilisation comme le cadrage de la vidéo, la visibilité des mains ainsi que l'exécution correcte de l'exercice, puisqu'ils sont des facteurs déterminants pour la crédibilité des résultats. Travailler avec les corrélations semble la solution la plus intéressante dans le cadre de comparaison des mouvements. Cependant, des manières différentes de procéder, notamment en tenant aussi compte de l'axe X et pas seulement du Y pourraient être envisageable si des connaissances expertes en statistiques permettaient de trouver un moyen de les corréler.

En ce qui concerne l'application web, la solution la plus durable serait d'obtenir du matériel physique de traitement d'images dans son propre serveur. L'allocation de la mémoire pour une exécution simultanée par plusieurs utilisateurs reste encore une problématique, mais un système de mise en attente selon la disponibilité du GPU pourrait résoudre ce point. Une piste alternative, pour garantir au mieux un accès à tout moment et donc sécuriser la disponibilité des systèmes, serait de trouver un nouveau distributeur de service.

Les résultats des analyses d'une vidéo effectuée sur l'application web ne sont accompagnés d'aucune garantie. Les graphiques et les mesures obtenus dépendent de la manière dont la vidéo a été prise selon les règles décidées pour la prise d'images. Même si toutes les conditions sont réunies pour une bonne quantification, l'utilisateur doit toujours garder un œil critique sur les résultats. En effet, il s'agit d'une application expérimentale et les graphiques servent d'outils pour comprendre comment les mouvements ont été suivis. Il est également important de vérifier la vidéo de sortie du logiciel pour s'assurer que le suivi des mouvements a été effectué correctement.

Une phase de tests de l'application afin d'en garantir l'évolutivité et la fiabilité pourrait permettre son utilisation à plus large échelle, en dehors de ce cadre expérimental.

RÉFÉRENCES

Andreeli, F., & Mosbah, H. (2014, Février). IRM fonctionnelle cérébrale : les principes. *Médecine des Maladies Métaboliques, 8*(1), 13-19. doi:10.1016/s1957-2557(14)70677-7

Bahramudin. (2019, Janvier 25). *Error : Failed to get convolution algorithm. This is probably because cuDNN failed to initialize, so try looking to see if a warning log message was printed above. #24828*, [Publication sur forum]. Consulté le Juillet 28, 2021, sur https://github.com/tensorflow/tensorflow/issues/24828

Bazel. (2021, Février 19). *Installing Bazel on Windows*. Consulté le Juillet 31, 2021, sur Bazel: https://docs.bazel.build/versions/main/install-windows.html

Brownlee, J. (2018, Juillet 20). *Difference Between a Batch and an Epoch in a Neural Network*. Consulté le Juillet 14, 2021, sur Machine Learning Mastery: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Carr, L., Harrison, L., Evans, A., & Stephens, J. (1993, Octobre 1). Patterns of central motor reorganization in hemiplegic cerebral palsy. *Brain, 116*(5), pp. 1233-1247. doi:10.1093/brain/116.5.1223

Cheong, J. (2020, Décembre 8). *Four ways to quantify synchrony between time series data*. doi:10.17605/OSF.IO/BA3NY

Connolly, K., & Stratton, P. (1968). Developmental Changes in Associated Movements. *Developmental Medicine & Child Neurology, 10*(1), pp. 49-56. doi:10.1111/j.1469-8749.1968.tb02837.x

De Winter, D., Klingels, K., & Simon Martinez, C. (2016). *Mirror Movements in Typically Developing Children.* Leuven: KU Leuven. Faculteit Bewegings- en Revalidatiewetenschappen. Récupéré sur https://limo.libis.be/permalink/f/qhi5mr/32LIBIS_ALMA_DS71187148140001471

deadcode. (2019, Avril 4). *What's the difference between track and tracklet in tracking?* Consulté le Juillet 18, 2021, sur Stack Overflow: https://stackoverflow.com/questions/55512548/whats-the-difference-between-track-and-tracklet-in-tracking

DeepLabCut. (2020a, Avril 15). *DeepLabCut - Skeleton Builder.* Consulté le Juillet 14, 2021, sur YouTube: https://www.youtube.com/watch?v=fQSJ0S08UyY

DeepLabCut. (2020b, May 3). *DeepLabCut Project Manager GUI.* Consulté le Juillet 16, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/blob/0e5380bb04d49a7ab97427c0d15173bb88c7559a/docs/functionDetails.md

DeepLabCut. (2021a, Juillet 1). *How To Install DeepLabCut*. Consulté le Juillet 14, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/blob/master/docs/installation.md

DeepLabCut. (2021b, Juillet 1). *Releases.* Consulté le Juillet 13, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/releases

DeepLabCut. (2021c, Mai 2). *Documentation Overview.* Consulté le Juillet 13, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/blob/master/docs/UseOverviewGuide.md#important-information-on-using-deeplabcut

DeepLabCut. (2021d, Juillet 1). *Welcome to Our Documentation Hub!* Consulté le Juillet 13, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/blob/master/docs/README.md

DeepLabCut. (2021e, Juillet 5). *DeepLabCut for Multi-Animal Projects.* Consulté le Juillet 13, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/blob/master/docs/maDLC_UserGuide.md

DeepLabCut. (2021f, Mai 3). *DeepLabCut Blog*. Consulté le Juillet 15, 2021, sur Mathis Laboratory of adaptive motor control: http://www.mackenziemathislab.org/deeplabcutblog/tag/deeplabcut

DeepLabCut. (2021g, Juillet 16). *DeepLabCut Model Zoo!* Récupéré sur Mathis Laboratory of adaptive motor control: http://www.mackenziemathislab.org/dlc-modelzoo

DeepLabCut. (2021h, Mai 27). *DeepLabCut 2.2 Toolbox - COLAB.* Consulté le Juillet 18, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/blob/master/examples/COLAB_maDLC_TrainNetwork_VideoAnalysis.ipynb

DeepLabCut. (2021i, Juin 11). *DLC-GPU-LITE.yaml*. Consulté le Juillet 28, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/blob/20572ba99559a085d968bd2a8ca49baf240037be/conda-environments/DLC-GPU-LITE.yaml

DeepLabCut. (2021j, Juillet 1). *DeepLabCut User Guide (for single animal projects)*. Consulté le Juillet 31, 2021, sur GitHub: https://github.com/DeepLabCut/DeepLabCut/commits/master/docs/standardDeepLabCut_UserGuide.md

Dinomais, M., Hertz-Pannier, L., & Nguyen The Tich, S. (2014, Février 6). Réorganisation du cortex sensorimoteur dans le cadre de la paralysie cérébrale unilatérale : apports des neurosciences. *Motricité Cérébrale : Réadaptation, Neurologie du Développement, 35*(1), 3-14. doi:10.1016/j.motcer.2013.12.001

EFTA. (2018, Juillet 19). *General Data Protection Regulation (GDPR) entered into force in the EEA*. Consulté le Août 1, 2021, sur EFTA: https://www.efta.int/EEA/news/General-Data-Protection-Regulation-GDPR-entered-force-EEA-509576

Evanson, N. (2020, Décembre 28). *Explainer: What are MMX, SSE, and AVX?* Consulté le Juillet 31, 2021, sur TECHSPOT: https://www.techspot.com/article/2166-mmx-sse-avx-explained/

Fritz Labs Incorporated. (2021). *Pose Estimation Guide*. Consulté le Juin 15, 2021, sur Fritz AI: https://www.fritz.ai/pose-estimation/

Galléa, C., Popa, T., Billot, S., Méneret, A., Depienne, C., & Roze, E. (2011, Juin 3). Congenital mirror movements: a clue to understanding bimanual motor control. *Journal of Neurology*(258), pp. 1911-1919. doi:10.1007/s00415-011-6107-9

Google. (s.d.). *Bienvenue dans Colaboratory*. Consulté le Juillet 2, 2021, sur Google Colaboratory: https://colab.research.google.com/notebooks/intro.ipynb

Graving, J., Chae, D., Naik, H., Liang, L., Koger, B., Costelloe, B., & Couzin, I. (2019, Octobre 1). DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife, 8*, pp. 1-42. doi:10.7554/elife.47994

Hunnicutt, B., Long, B., Kusefoglu, D., Gertz, K., Zhong, H., & Mao, T. (2014, Septembre). A comprehensive thalamocortical projection map at the mesoscopic level. *Nature neuroscience, 17*(9), pp. 1276-1285. doi:10.1038/nn.3780

jakes. (2019, Octobre 15). *Jupyter Lab freezes the computer when out of RAM - how to prevent it?* Consulté le Août 1, 2021, sur Stack Overflow: https://stackoverflow.com/questions/58400437/jupyter-lab-freezes-the-computer-when-out-of-ram-how-to-prevent-it

Jaspers, E., Byblow, W., Feys, H., & Wenderoth, N. (2016, Janvier 6). The Corticospinal Tract: A Biomarker to Categorize Upper Limb Functional Potential in Unilateral Cerebral Palsy. *Frontiers in Pediatrics, 3*, 1-10. doi:doi.org/10.3389/fped.2015.00112

Jaspers, E., Klingels, K., Simon-Martinez, C., Feys, H., Woolley, D., & Wenderoth, N. (2017, Juillet 5). GriFT: A Device for Quantifying Physiological and Pathological Mirror Movements in Children. *IEEE Transactions on Biomedical Engineering, 65*(4), 857-865. doi:10.1109/TBME.2017.2723801

Klingels, K., Jaspers, E., Staudt, M., Guzzetta, A., Mailleux, L., Ortibus, E., & Feys, H. (2015, Décembre 8). Do mirror movements relate to hand function and timing of the brain lesion in children with unilateral cerebral palsy? *Developmental Medicine & Child Neurology (DMCN), 58*(7), pp. 735-742. doi:10.1111/dmcn.12977

Koerte, I., Eftimov, L., Laubender, R., Esslinger, O., Schroeder, A., Ertl-Wagner, B., . . . Danek, A. (2010, Octobre 11). Mirror movements in healthy humans across the lifespan: effects of development and ageing. *Developmental Medicine & Child Neurology (DMCN)*, pp. 1106-1112. doi:10.1111/j.1469-8749.2010.03766.x

Koerte, I., Pelavin, P., Kirmess, B., Fuchs, T., Berweck, S., Laubender, R., . . . Heinen, F. (2011). Anisotropy of transcallosal motor fibres indicates functional impairment in children with periventricular leukomalacia. *Developmental Medicine & Child Neurology, 53*(2), pp. 179-186. doi:10.1111/j.1469-8749.2010.03840.x

Kuhtz-Buschbeck, J., Krumlinde-Sundholm, L., Eliasson, A.-C., & Forssberg, H. (2007, Février 13). Quantitative assessment of mirror movements in children and adolescents with hemiplegic cerebral palsy. *Developmental Medicine & Child Neurology, 42*(11), 728-736. doi:10.1111/j.1469-8749.2000.tb00034.x

Kuo, H.-C., Friel, K., & Gordon, A. (2017, Septembre 8). Neurophysiological mechanisms and functional impact of mirror movements in children with unilateral spastic cerebral palsy. *Developmental Medicine & Child Neurology, 60*, pp. 155-161. doi:10.1111/dmcn.13524

Lauer, J., Zhou, M., Ye, S., Menegas, W., Nath, T., Rahman, M., . . . Mathis, A. (2021, Avril 30). Multi-animal pose estimation and tracking with DeepLabCut. *bioRxiv*, pp. 1-20. doi:10.1101/2021.04.30.442096

Liu, X., Yu, S.-y., Flierman, N., Loyola, S., Kamermans, M., Hoogland, T., & Zeeuw, C. (2021, Mai 28). OptiFlex: Multi-Frame Animal Pose Estimation Combining Deep Learning With

Optical Flow. *Frontiers in Cellular Neuroscience, 15*, pp. 1-14. doi:10.3389/fncel.2021.621252

Magne, V., Adde, L., Hoare, B., Klingels, K., Simon-Martinez, C., Mailleux, L., . . . Elvrum, A.-K. (2021, Janvier 19). Assessment of mirror movements in children and adolescents with unilateral cerebral palsy: reliability of the Woods and Teuber scale. *Developmental Medicine & Child Neurology (DMCN)*, pp. 736-742. doi:10.1111/dmcn.14806

Mailleux, L., Simon-Martinez, C., Klingles, K., Ortibus, E., & Feys, H. (2021, Janvier 1). Brain lesion characteristics in relation to upper limb function in children with unilateral cerebral palsy. *Factors Affecting Neurodevelopment*, pp. 411-420. doi:10.1016/B978-0-12-817986-4.00035-3

Mathis, A., & Warren, R. (2018, Janvier 1). On the inference speed and video-compression robustness of DeepLabCut. *bioRxiv*, pp. 1-10. doi:10.1101/457242

Mathis, A., Mamidanna, P., Cury, K., Abe, T., Murthy, V., & Mathis, M. (2018, Août 20). DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience, 21*, pp. 1281-1289. doi:10.1038/s41593-018-0209-y

Mathis, A., Schneider, S., Lauer, J., & Mathis, M. (2020, Septembre 2). A Primer on Motion Capture with Deep Learning: Principles, Pitfalls and Perspectives. *Neuron, 108*(1), pp. 44-65. doi:10.1016/j.neuron.2020.09.017

Ng, H. (2021). *How to Create an Interactive Web Application using a Jupyter Notebook*. Récupéré sur Finxter: https://blog.finxter.com/how-to-create-an-interactive-web-application-using-jupyter-notebook/

Odemakinde, E. (2021, Avril 16). *State of the Art Pose Estimation: The Ultimate Overview*. Consulté le Mai 26, 2021, sur viso.ai: https://viso.ai/deep-learning/pose-estimation-ultimate-overview/

Ortega, M. (2019, Juillet 10). *Into the problem of Hand Recognition.* Consulté le Juillet 15, 2021, sur Medium: https://ortegatron.medium.com/into-the-problem-of-hand-recognition-da30797450fe

Ortega, M. (2020, Février 27). *Training a Hand Detector like the OpenPose one in Tensorflow.* Consulté le Juillet 15, 2021, sur Medium: https://ortegatron.medium.com/training-a-hand-detector-like-the-openpose-one-in-tensorflow-45c5177d6679

pandas. (2021, Juillet 2). *pandas - Python Data Analysis Library*. Consulté le Juillet 23, 2021, sur pandas: https://pandas.pydata.org/

Paperspace. (2021). *Gradient - Effortless Deep Learning at scale*. Consulté le Juillet 27, 2021, sur Gradient by Paperspace: https://gradient.paperspace.com/

Project Jupyter. (2020, Janvier 7). *Layout and Styling of Jupyter widgets*. Consulté le Juillet 30, 2021, sur Jupyter Widgets: https://ipywidgets.readthedocs.io/en/stable/examples/Widget%20Styling.html

Project Jupyter. (2021, Juillet 13). *Jupyter*. Consulté le Juillet 20, 2021, sur Jupyter: https://jupyter.org/

Python Software Foundation. (2021, Juillet 31). *pickle — Python object serialization*. Consulté le Juillet 31, 2021, sur Python: https://docs.python.org/3/library/pickle.html

Robert, C. (2021, Février 11). *What is the correct version of CUDA for my nvidia driver?* Consulté le Juillet 28, 2021, sur Stack Overflow: https://stackoverflow.com/questions/30820513/what-is-the-correct-version-of-cuda-for-my-nvidia-driver/30820690#30820690

Schmitt, M. (2020). *Data dashboarding tools | Streamlit vs. Dash vs. Shiny vs. Voila vs. Flask vs. Jupyter*. Consulté le Juillet 27, 2021, sur datarevenue: https://www.datarevenue.com/en-blog/data-dashboarding-streamlit-vs-dash-vs-shiny-vs-voila

Simon-Martinez, C., Zielinksi, I., Hoare, B., Decraene, L., Williams, J., Mailleux, L., . . . Klingels, K. (2020, Février 3). The impact of brain lesion characteristics and the corticospinal tract wiring pattern on mirror movement characteristics in unilateral cerebral palsy. *medRxiv*, 1-28. doi:10.1101/2020.01.31.20019893

Sprung, J. (2020, Mai 20). *How to configure Instances via Startup Scripts*. Consulté le Juillet 28, 2021, sur Genesis Cloud: https://support.genesiscloud.com/support/solutions/articles/47001122478

Staudt, M. (2010, Juillet 23). Reorganization after pre- and perinatal brain lesions*. *Journal of Anatomy, 217*(4), pp. 469-474. doi:10.1111/j.1469-7580.2010.01262.x

Tanmay, N., Mathis, A., Chen, A., Patel, A., Bethge, M., & Mathis, M. (2019, Juin 21). Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols, 14*, pp. 2152-2176. doi:10.1038/s41596-019-0176-0

The MathWorks, Inc. (s.d.). *Segmentation sémantique*. Consulté le Mai 31, 2021, sur MathWorks: https://ch.mathworks.com/fr/solutions/image-video-processing/semantic-segmentation.html

Tisseyre, J. (2019, Décembre). Corrélats comportementaux, neurophysiologiques et neuropsychologiques des mouvements miroirs chez le sujet sain et le patient cérébro-lésé. *Neurosciences [q-bio.NC]*, 1-230. Consulté le Juin 15, 2021, sur https://tel.archives-ouvertes.fr/tel-02934221/document

Venkatesh, P., Xu, J., & Tsai, H.-J. (2021, Juillet 26). *Intel® Optimization for TensorFlow* Installation Guide*. Consulté le Juillet 31, 2021, sur Intel Corporation: https://software.intel.com/content/www/us/en/develop/articles/intel-optimization-for-tensorflow-installation-guide.html#collapseCollapsible1627320337186

Welniarz, Q., Dusart, I., Gallea, C., & Roze, E. (2015, Juin 2). One hand clapping: lateralization of motor control. *Frontiers in Neuroanatomy, 9*, 1-75. doi:doi.org/10.3389/fnana.2015.00075

Woods, B., & Teuber, H.-L. (1978, Novembre 1). Mirror movements after childhood hemiparesis. *Neurology, 28*(11), pp. 1152-1157. doi:10.1212/WNL.28.11.1152

Zielinksi, I., Steenbergen, B., Schmidt, A., Klingels, K., Simon Martinez, C., de Water, P., & Hoare, B. (2018, Mai 26). Windmill-task as a New Quantitative and Objective Assessment for Mirror Movements in Unilateral Cerebral Palsy: A Pilot Study. pp. 1547-1552. doi:10.1016/j.apmr.2018.01.035

Annexe I : Tableau comparatif de logiciels d'estimation de pose

Tableau : Programmes d'estimation de pose en 2D avec reconnaissance d'image possible pour l'humain

Logiciel	Toutes espèces d'animaux	Multi animal	GUI	Données d'exemple	Documentation	Communauté	Citations[36]
DeepBehaviour[a]	Non	Oui	Partiellement	Non	Guide	GitHub	37
DeepLabCut[b]	Oui	Oui	Oui	Oui	Guides Workshop	Forum GitHub Chat	838
DeepPoseKit[c]	Oui	Non	Partiellement	Oui	Guide	GitHub	138
Optiflex[e]	Oui	Oui	Partiellement	Oui	Guide	GitHub	8
SLEAP[f]	Oui	Oui	Oui	Oui	Guides	GitHub	195[37]

Source : Tableau de l'auteur provenant de sources multiples

a. Arac, A., Zhao, P., Dobkin, B. H., Carmichael, S. T., & Golshani, P. (2019). DeepBehavior: A Deep Learning Toolbox for Automated Analysis of Animal and Human Behavior Imaging Data. Frontiers in Systems Neuroscience, 13. doi:10.3389/fnsys.2019.00020

b. Mathis, A., Mamidanna, P., Cury, K.M. et al. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nat Neurosci 21, 1281–1289 (2018). doi:10.1038/s41593-018-0209-y

c. Graving, J. M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B. R., & Couzin, I. D. (2019). DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. eLife, 8. doi:10.7554/elife.47994

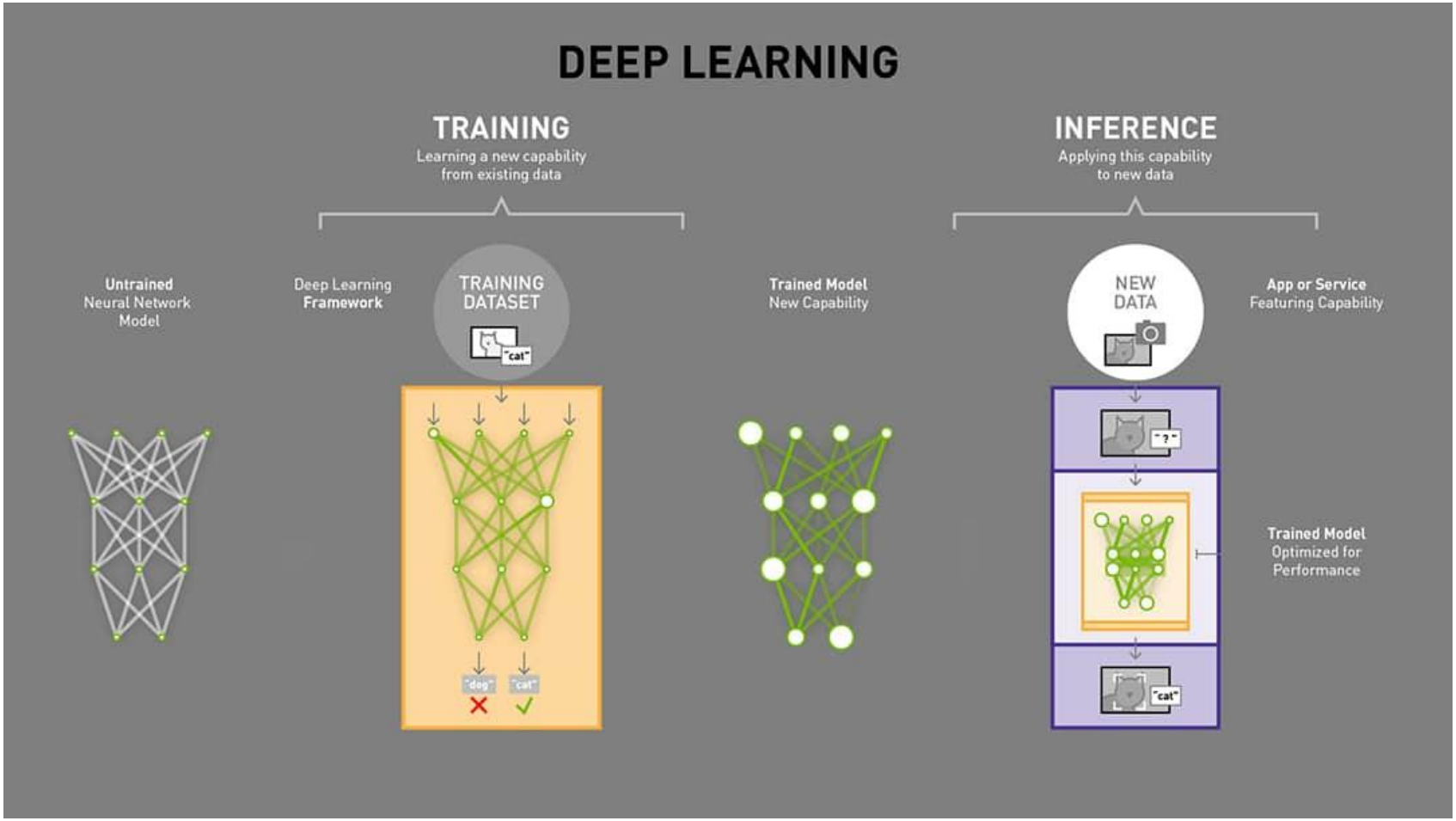
[36] Basé sur les chiffres fournis par Google Scholar le 2 juillet 2021.

[37] Chiffre basé sur le rapport scientifique de LEAP, le prédécesseur de SLEAP

d. Liu, X., Yu, S. Y., Flierman, N., Loyola, S., Kamermans, M., Hoogland, T. M., & De Zeeuw, C. I. (2020). OptiFlex : video-based animal pose estimation using deep learning enhanced by optical flow. bioRxiv. doi:10.1101/2020.04.04.025494

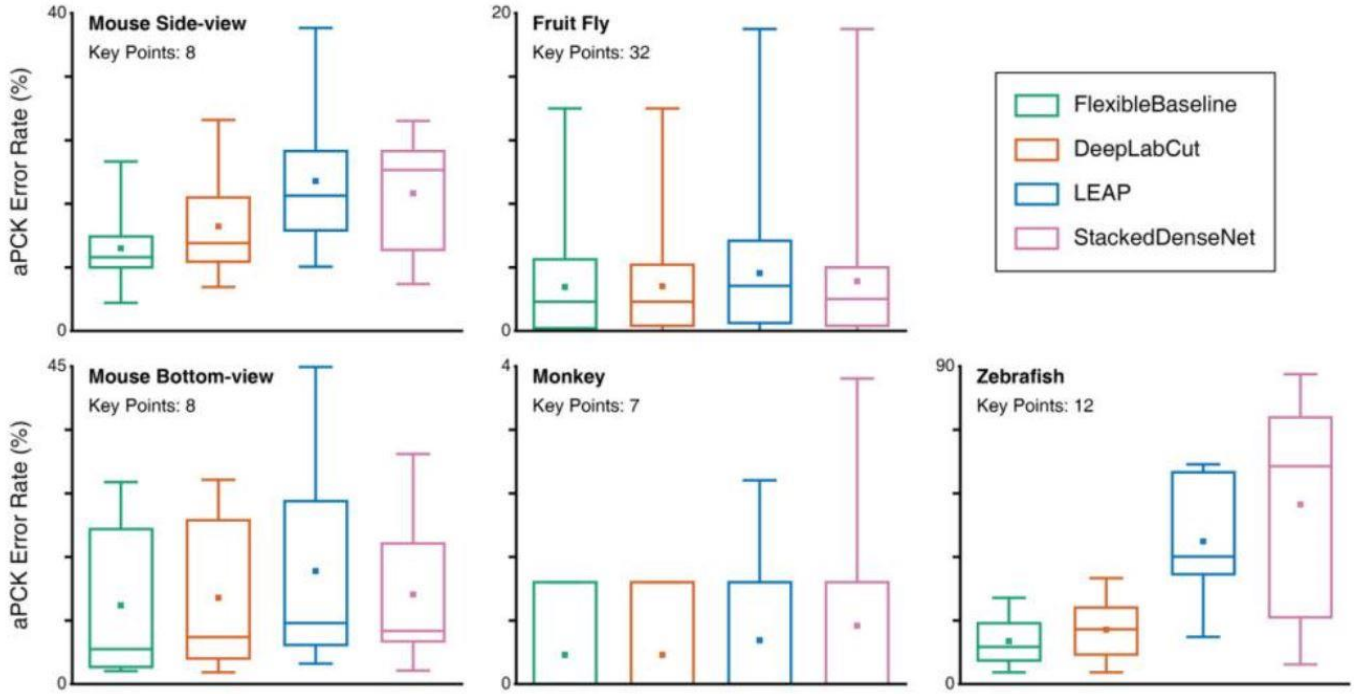
e. Pereira, T.D., Aldarondo, D.E., Willmore, L. et al. Fast animal pose estimation using deep neural networks. Nat Methods 16, 117–125 (2019). doi:10.1038/s41592-018-0234-5

Annexe II : Différenciation entre l'entraînement et l'inférence



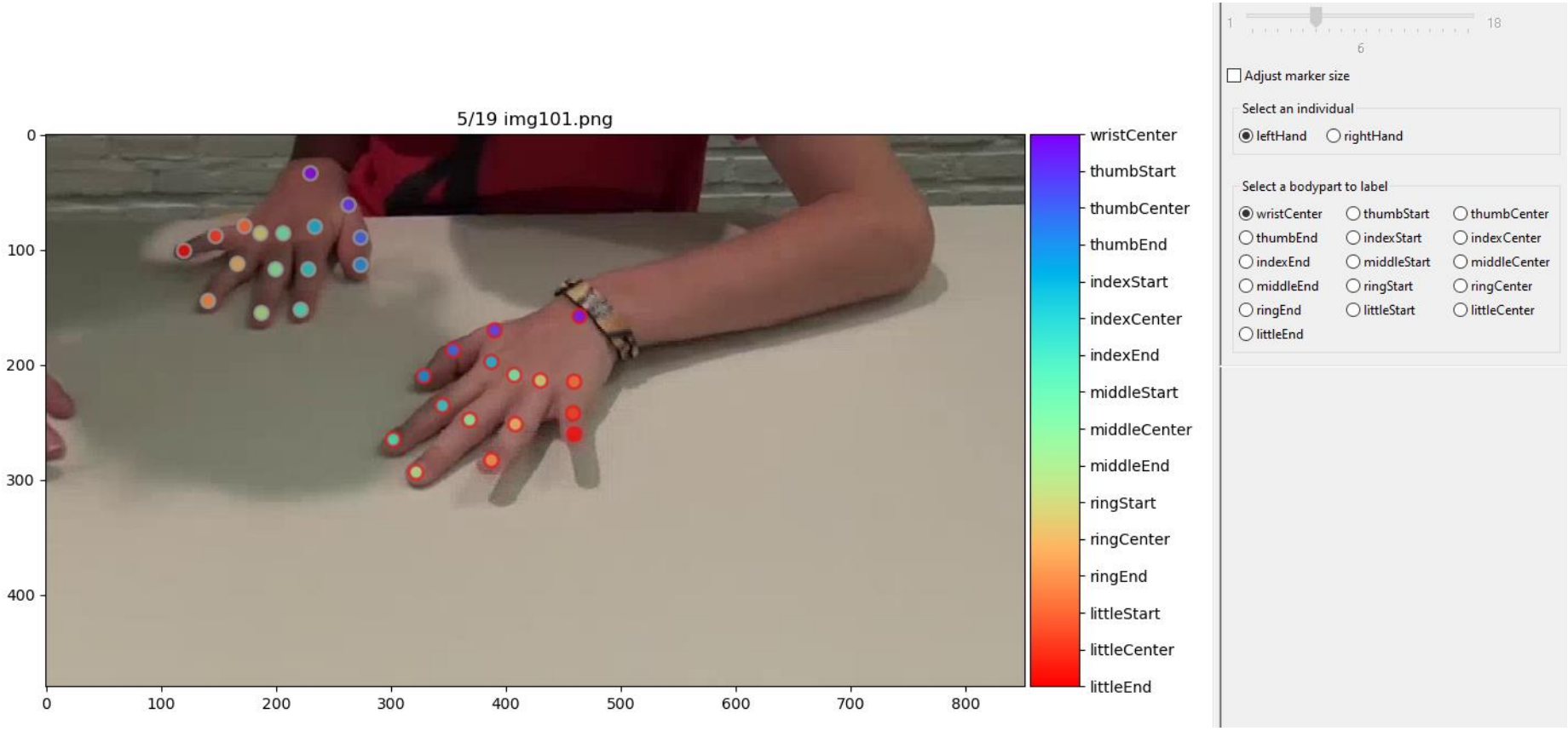
Source : Copeland, M. (2019). What's the Difference Between Deep Learning Training and Inference? Récupéré le 5 juillet 2021 sur https://blogs.nvidia.com/blog/2016/08/22/difference-deep-learning-training-inference-ai/

Annexe III : Comparaison des performances d'algorithmes



Source : Liu, et al. (2021). OptiFlex : Multi-Frame Animal Pose Estimation Combining Deep Learning With Optical Flow. Frontiers in Cellular Neuroscience, 15, 9. https://doi.org/10.3389/fncel.2021.621252

Annexe IV : Interface graphique de labélisation



Source : Capture d'écran de l'auteur

Annexe V : Prix indicatifs des fournisseurs de *cloud* GPU

Tableau : Comparaison des prix indicatifs des fournisseurs de GPU dans le cloud

Service	Genesis Cloud[a]	Paperspace[b]	Puzl.ee[c]	Saturn Cloud[d]
API	Oui	Oui	Oui	Oui
GPU	NVIDIA 1080 TI	NVIDIA P4000	NVIDIA 2080Ti	T4-XLarge
Giga-octets (GO) dédiés	11 GO (GDDR5X)	8 GO (GDDR5)	11 GO (GDDR6)	16 GO (?)
Prix par heure	~0,55 CHF	~0,47 CHF	~0,36 CHF	~0,63 CHF

Source : Tableau de l'auteur provenant de sources multiples

a. Genesis Cloud. (2021). Genesis Cloud - Pricing. Consulté le 1 août 2021, à l'adresse https://www.genesiscloud.com/pricing/

b. Paperspace. (2020). Pricing. Consulté le 1 août 2021, à l'adresse https://www.paperspace.com/pricing

c. Puzl.ee. (2021). Compute Resources. Consulté le 1 août 2021, à l'adresse https://puzl.ee/cloud-kubernetes/computing-resources

d. Saturn Cloud. (2021, 12 juillet). Saturn Hosted Pro pricing | Saturn Cloud. Consulté le 1 août 2021, à l'adresse https://saturncloud.io/plans/hosted_pro_pricing/

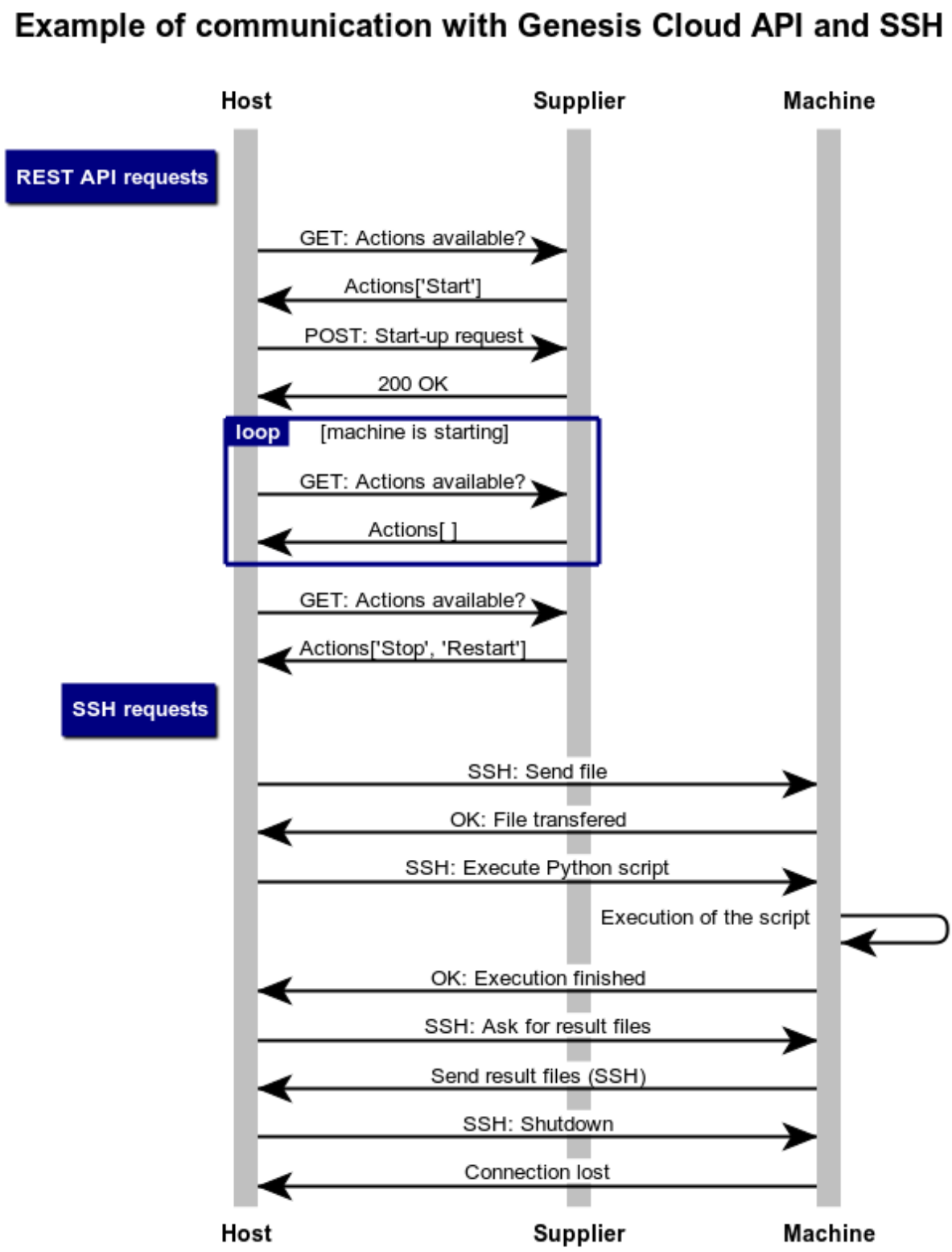
Annexe VI : Product backlog

Disponible en accès complet sur https://bit.ly/3xftayM

US Nr.	Theme	User Stories			Acceptance Criteria	Priority	Status	Story Points	Sprint	US accepted (done done)	MoSCoW
		As an/a ...	I want to ...	so that ...							
7	Software	User	See the output video of the analysis	It is possible to see if it the hands were succesfully identified by the algorithm	See the output video of the algorithm	1000		21	0	07.06.2021	Must
4	Software	User	Have an analysis of the amplitude	I will have an idea of how important is the amplitude	Analysis of the amplitude	990		21	1	-	Must
10	Software	User	Have an analysis of the amplitude with a specific exercise	I will have an idea of how important is the amplitude and the correlation	Analysis of the amplitude with specific exercise defined	950		21	2	19.07.2021	Must
11	Software	User	Have some visual reprensatations of the tracking	I can see the variation	Some graphs are available to show the movements during the analyzed video	920		2	2	19.07.2021	Must
1	GUI	User	Upload the video online	I do not need any specific material other than a camera to do an analysis	User are capable of upload videos on the platform	800		21	3	02.08.2021	Must
6	GUI	User	See the evaluation of the mirror movements	I can have results of the analysis	Having GUI to see the measurements	780		3	3	02.08.2021	Must
2	GUI	User	Download the result from the analysis	I can store them localy or in the patient file	User can download the analysis localy	760		3	3	02.08.2021	Must
3	Software	User	Have graphical representation of the correlation	I can see visually the correlation	Graph with correlation available	550		2	3	02.08.2021	Must
9	GUI	User	Have documentation on how the application work and how to use it	I am to work correctly and understand how the application works	A small documentation is available to the user	500		1	3	02.08.2021	Must
5	Software	User	Have an analysis of the frequency	I will have an idea of how important is the frequency	Analysis of the frequency	420		13	-		Should
8	GUI	User	Be able to choose the segment I want to work with	It is possible to use my own segmentation	The user can choose the segment before doing the video analysis	400		34	-		Could
						Total SP		142			

Source : Données de l'auteur

Annexe VII : Schéma de communication pour un serveur externe



Source : Schéma de l'auteur

Annexe VIII : Documentation sur la prise d'image

Source : Document de l'auteur.

Detect and quantify mirror movements in videos of children with cerebral palsy using DeepLabCut

Documentation on the use of the application and video specifications

1. CONTEXT

The estimation of the pose by inference as well as the quantification of the mirror movements are performed with a video. The video should contain a selected exercise with defined conditions.

2. EXERCISE AND VIDEO

The exercise consists of placing the hands on a table, raising, and laying each finger of one hand in turn while the other hand remains at rest. For the detection and tracking conditions to be optimal, the following rules must be observed:

- The video should be as stable as possible but does not necessarily require any accessories such as a tripod and can be filmed using a smartphone;
- The video should not be longer than 10 seconds, which is more than enough time to do this exercise calmly;
- The camera angle[1] should be about 35 degrees (Image 1, see example 3.2);
- Both hands should be in focus throughout the video;
- Hands should be at the same height on the table (See example 3.1);
- The video should be filmed in landscape and the middle fingers of the hand should be on the vertical axis.

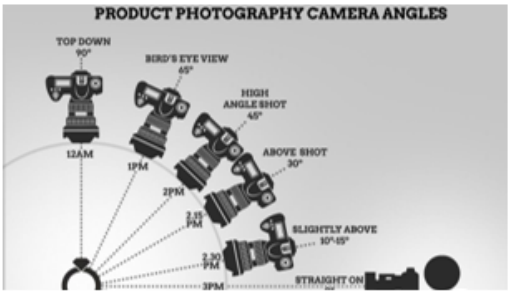


Image 1 : Different angles for video and photography
Source: https://id.pinterest.com/pin/297026537903737069/

[1] Good angle means that throughout the exercise the nails are always visible to the camera and the range of motion can be correctly seen.

Page 1 of 3

Detect and quantify mirror movements in videos of children with cerebral palsy using DeepLabCut

3. EXAMPLES

3.1. ALIGNMENT OF HANDS

3.2. FRAMING ANGLE OF THE VIDEO

3.3. VERTICAL AXIS ALIGNMENT OF FINGERS

4. WARRANTY

This application does not come with any warranty. Graphics and measures depend on how the video was taken according to this document. Even if all the conditions are met for a good quantification, the user must always keep a critical eye on the results. Indeed, this is an experimental application, and the graphs serve as a tool to understand how the movements were tracked. It is important to check the output video of the software to ensure that the motion tracking has been performed correctly.

Déclaration de l'auteur

Je déclare, par ce document, que j'ai effectué le travail de bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- Monsieur Henning Müller, professeur responsable du suivi de ce travail ;
- Madame Cristina Simon-Martinez, mandante et référente technique.

Sierre, le 4 août 2021

Brice Berclaz