NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Human action prediction from hand movement for human-robot collaboration

Nikolaos K. Soulounias

Supervisors: **Maria Dagioglou**, National Center for Scientific Research (NSCR) “Demokritos”, Associate Researcher

Panagiotis Stamatopoulos, National and Kapodistrian University of Athens (NKUA), Assistant Professor

ATHENS
AUGUST 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη δράσης ανθρώπου από την κίνηση του χεριού για συνεργασία ανθρώπου-ρομπότ

Νικόλαος Κ. Σουλούνιας

Επιβλέποντες: **Μαρία Δαγιόγλου**, Εθνικό Κέντρο Έρευνας Φυσικών Επιστημών (ΕΚΕΦΕ) «Δημόκριτος», Συνεργάτης Ερευνήτρια

Παναγιώτης Σταματόπουλος, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών (ΕΚΠΑ), Επίκουρος Καθηγητής

ΑΘΗΝΑ
ΑΥΓΟΥΣΤΟΣ 2021

BSc THESIS

Human action prediction from hand movement for human-robot collaboration

Nikolaos K. Soulounias
S.N.: 1115201600156

Supervisors: **Maria Dagioglou**, National Center for Scientific Research (NSCR) "Demokritos", Associate Researcher

Panagiotis Stamatopoulos, National and Kapodistrian University of Athens (NKUA), Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη δράσης ανθρώπου από την κίνηση του χεριού για συνεργασία ανθρώπου-ρομπότ

Νικόλαος Κ. Σουλούνιας
Α.Μ.: 1115201600156

Επιβλέποντες: **Μαρία Δαγιόγλου**, Εθνικό Κέντρο Έρευνας Φυσικών Επιστημών (ΕΚΕΦΕ) «Δημόκριτος», Συνεργάτης Ερευνήτρια

Παναγιώτης Σταματόπουλος, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών (ΕΚΠΑ), Επίκουρος Καθηγητής

ABSTRACT

Human-Robot Collaboration can combine the speed and precision of a robot with the cognitive abilities of a human. In a Human-Robot Collaboration scenario, the human and the robot agent share the same workspace and coordinate their movements in order to achieve a common goal. Therefore, the robot's ability to predict the action of the human before its completion is crucial to ensure safe and fluent collaboration between the agents. For example, in a production line the human agent's actions involve grasping different objects. In this case we would like for the robot to be able to predict the object to be grasped.

This thesis focuses on the task of predicting the object to be grasped based solely on the human hand movement without considering the object's properties. The core assumption of our work relied on behavioral findings that show that the hand gradually conforms to the properties of the target object throughout the entire grasping movement, not only during the end of it. Such results were obtained using body markers and a high-end motion capture system of multiple cameras. However, in a Human - Robot Collaboration set-up, the use of a single RGB-D camera would be desired. Unfortunately, this comes at the cost of an increased amount of noise in the hand pose information. The goal of this thesis was to explore whether it is possible to predict the size of an object based solely on human arm and hand kinematics obtained through a single RGB-D camera.

In our work, we used a dataset that included movements towards three objects with the same cubical shape, but different size. The dataset was collected using an RGB-D visual sensor. The OpenPose framework was chosen for the human arm and hand pose estimation. The keypoints detections were 2D. After the movements were preprocessed to filter out noisy frames, identify the grasping movement and select a part of the grasping movement, the kinematic features of this partial grasping movement were engineered. The features which were investigated were the fingertip aperture, the wrist coordinate, the wrist instantaneous speed and the wrist coordinates' dispersion features. Following that, both "traditional" Machine Learning models and a Deep Learning model were used. The "traditional" Machine Learning models included Random Forest, Gradient Boosting, Extra Trees, Support Vector Machine and Gaussian Process, while the Deep Learning model was a simple Convolutional Neural Network. The models were evaluated for different kinematic feature combinations, different training set - cross-validation/testing set partition strategies and different movement completion percentages. The timing of the predictions and the performance of the best models was also evaluated for a simple HRC application using real robot data.

Due to the dataset's size, the top-performing "traditional" Machine Learning models outperformed the Deep Learning model consistently. Furthermore, the models' accuracy was higher when they were trained with movements of all the participants as opposed to when they predicted the movements of a participant who did not belong in the training set. For the former case the best accuracy was logged by the Extra Trees model and was 66.27%, 80.14%, 86.01%, 91.88% and 93.84% for 20%, 40%, 60%, 80% and 100% of the movement respectively. For the latter case the best accuracy was logged by the Support Vector Machine model and was 50.83%, 61.67%, 66.58%, 64.93% and 69.08% for the same movement completion percentages. The reason for this phenomenon was that a Machine Learning model was not able to generalize from the training set's participants to the testing set's participant, if the latter's movement pattern was significantly different from the other participants' movement patterns. Finally, the Machine Learning models learned to identify the small object as non-large and the large

object as non-small. On the contrary, the medium class was mainly responsible for the suboptimal performance of the models.

SUBJECT AREA: Human-Robot Collaboration, Machine Learning, Computer Vision, Skeleton-based Human Action Prediction

KEYWORDS: Hand Skeletal Data, Kinematic Features, Prediction

ΠΕΡΙΛΗΨΗ

Η συνεργασία ανθρώπου-ρομπότ μπορεί να συνδυάσει την ταχύτητα και την ακρίβεια ενός ρομπότ με τις γνωστικές ικανότητες ενός ανθρώπου. Σε ένα σενάριο συνεργασίας ανθρώπου-ρομπότ, ο άνθρωπος και το ρομπότ μοιράζονται τον ίδιο χώρο εργασίας και συντονίζουν τις κινήσεις τους προκειμένου να επιτύχουν έναν κοινό στόχο. Ως εκ τούτου, η ικανότητα του ρομπότ να προβλέπει την ενέργεια του ανθρώπου πριν από την ολοκλήρωσή της είναι ζωτικής σημασίας για να διασφαλιστεί η ασφαλής και απρόσκοπτη συνεργασία μεταξύ των δύο συνεργατών. Για παράδειγμα, σε μια γραμμή παραγωγής, οι ενέργειες του ανθρώπου αφορούν το πιάσιμο διαφορετικών αντικειμένων. Στην περίπτωση αυτή θα θέλαμε το ρομπότ να μπορεί να προβλέψει το αντικείμενο το οποίο θα πιάσει ο άνθρωπος.

Αυτή η πτυχιακή εστιάζει στην πρόβλεψη του αντικειμένου το οποίο θα πιάσει ο άνθρωπος, λαμβάνοντας υπόψη αποκλειστικά την κίνηση του χεριού του ανθρώπου και όχι τις ιδιότητες του αντικειμένου. Η βασική υπόθεση της δουλειάς μας βασίστηκε σε συμπεριφορικά ευρήματα που φανερώνουν ότι το χέρι προσαρμόζεται στις ιδιότητες του αντικειμένου-στόχου σταδιακά κατά τη διάρκεια ολόκληρης της κίνησης, όχι μόνο κατά το τέλος της. Τα αποτελέσματα για αυτά τα συμπεριφορικά ευρήματα προέκυψαν χρησιμοποιώντας body markers και ένα ακριβό σύστημα motion capture με πολλαπλές κάμερες. Ωστόσο, στο πλαίσιο συνεργασίας ανθρώπου-ρομπότ, η χρήση μιας μόνο RGB-D κάμερας θα ήταν επιθυμητή. Δυστυχώς, αυτό έχει ως αποτέλεσμα περισσότερο θόρυβο στα σκελετικά δεδομένα του χεριού. Ο στόχος της πτυχιακής μας ήταν να διερευνήσουμε κατά πόσον είναι δυνατόν να προβλέψουμε το μέγεθος ενός αντικειμένου βασιζόμενοι μόνο στην πληροφορία της κίνησης του άνω άκρου και του χεριού του ανθρώπου, η οποία έχει προκύψει από μία μόνο RGB-D κάμερα.

Στη δουλειά μας, χρησιμοποιήσαμε ένα σύνολο δεδομένων το οποίο περιλάμβανε κινήσεις προς τρία αντικείμενα με το ίδιο κυβικό σχήμα, αλλά διαφορετικό μέγεθος. Το σύνολο δεδομένων είχε συλλεχθεί χρησιμοποιώντας ένα RGB-D οπτικό αισθητήρα. Το μοντέλο OpenPose επιλέχθηκε για την εκτίμηση της στάσης του άνω άκρου και του χεριού του ανθρώπου. Οι προβλέψεις των θέσεων των αρθρώσεων ήταν 2D. Κατά την προεπεξεργασία των κινήσεων, τα θορυβώδη καρέ της κίνησης φιλτραρίστηκαν, το στάδιο κατά το οποίο ο άνθρωπος κινείται προς το αντικείμενο εντοπίστηκε και ένα ποσοστό αυτού του σταδίου απομονώθηκε. Στη συνέχεια, παρήχθησαν τα κινηματικά χαρακτηριστικά για αυτό το ποσοστό της κίνησης. Τα κινηματικά χαρακτηριστικά που διερευνήθηκαν ήταν το άνοιγμα των ακροδακτύλων και οι συντεταγμένες, η στιγμιαία ταχύτητα και η διασπορά των συντεταγμένων του καρπού. Ακολούθως, χρησιμοποιήθηκαν «παραδοσιακά» μοντέλα Machine Learning και ένα μοντέλο Deep Learning. Τα «παραδοσιακά» μοντέλα Machine Learning περιλάμβαναν τα Random Forest, Gradient Boosting, Extra Trees, Support Vector Machine και Gaussian Process, ενώ το μοντέλο Deep Learning ήταν ένα απλό Convolutional Neural Network. Τα μοντέλα αξιολογήθηκαν για διαφορετικούς συνδυασμούς κινηματικών χαρακτηριστικών, διαφορετικές στρατηγικές διαχωρισμού των δεδομένων σε σύνολα εκπαίδευσης και ελέγχου και διαφορετικά ποσοστά ολοκλήρωσης της κίνησης. Ο χρόνος στον οποίο πρέπει να γίνουν οι προβλέψεις και η απόδοση των καλύτερων μοντέλων αξιολογήθηκε και για ένα απλό σενάριο συνεργασίας ανθρώπου-ρομπότ χρησιμοποιώντας πραγματικά ρομποτικά δεδομένα.

Λόγω του μεγέθους του συνόλου δεδομένων, τα πιο αποδοτικά “παραδοσιακά” μοντέλα Machine Learning είχαν σταθερά καλύτερα αποτελέσματα από το Deep Learning μοντέλο. Επιπλέον, η ακρίβεια των μοντέλων ήταν μεγαλύτερη όταν αυτά εκπαιδεύονταν με κινήσεις όλων των ανθρώπων εν αντιθέσει με όταν προέβλεπαν τις κινήσεις ενός

ανθρώπου ο οποίος δεν άνηκε στο σύνολο εκπαίδευσης. Για την πρώτη περίπτωση η καλύτερη απόδοση καταγράφηκε από το μοντέλο Extra Trees και ήταν 66.27%, 80.14%, 86.01%, 91.88% και 93.84% για το 20%, 40%, 60%, 80% και 100% της κίνησης αντίστοιχα. Για την δεύτερη περίπτωση η καλύτερη απόδοση καταγράφηκε από το μοντέλο Support Vector Machine και ήταν 50.83%, 61.67%, 66.58%, 64.93% και 69.08% για τα ίδια ποσοστά ολοκλήρωσης της κίνησης. Ο λόγος για αυτό το φαινόμενο ήταν ότι ένα Machine Learning μοντέλο δεν μπορούσε να γενικεύσει από τους ανθρώπους του συνόλου εκπαίδευσης στον άνθρωπο του συνόλου ελέγχου, εάν το μοτίβο της κίνησης του τελευταίου ήταν σημαντικά διαφορετικό από τα μοτίβα της κίνησης των άλλων ανθρώπων. Τέλος, τα Machine Learning μοντέλα μάθαιναν να αναγνωρίζουν το μικρό αντικείμενο ως μη-μεγάλο και το μεγάλο αντικείμενο ως μη-μικρό. Αντίθετα, η μεσαία κλάση ήταν κυρίως υπεύθυνη για την υποβέλτιστη απόδοση των μοντέλων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Συνεργασία Ανθρώπου-Ρομπότ, Μηχανική Μάθηση, Υπολογιστική Όραση, Πρόβλεψη Ανθρώπινων Ενεργειών βασισμένη σε σκελετικά δεδομένα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Σκελετικά δεδομένα του χεριού, Κινηματικά Χαρακτηριστικά, Πρόβλεψη

CONTENTS

LIST OF FIGURES

LIST OF IMAGES

LIST OF TABLES

1. INTRODUCTION

The first industrial robot was put in use in the beginning of the 1960s [42]. Only a decade later, the robotics technology started being massively adopted by the manufacturing industry to automate production. Industrial robots transformed the industry to such an extent that the integration of robotics technology is referred to as the *Third Industrial Revolution* [43]. Nevertheless, while these machines were largely automated, they still relied on human input and intervention.

Nowadays, the manufacturing industry is revolutionized once again to further enhance automation. *Industry 4.0* relies on the collection and interchange of data through sensors and IoT devices, which is then utilized by robots to work autonomously without the need of human presence [43]. During this *Fourth Industrial Revolution*, over the past decade the operational stock of industrial robots has almost tripled and currently more robots than ever are deployed (Fig. 1.1). The mass adoption of autonomous industrial robots can be interpreted as an effort to increase efficiency and productivity and to lower production costs.

The vast majority of robots that are currently deployed is preprogrammed to a significant degree for a specific task [42]. At the same time, the future goal of the manufacturing industry (*Industry 5.0* [47]) is the mass production of individually personalized products, instead of products focused on a target group. In order to achieve this future goal, Industry 5.0 will be focused on *Human-Robot Collaboration (HRC)*, instead of the full automation of the tasks. In this context, a collaborative robot, or cobot, will operate together with a human worker in the same workspace and will coordinate with its partnering agent to solve a task. The reason why this approach can support personalization is that human-robot collaboration combines both the advantages of robots and human workers. In particular, while human workers can adapt effectively to changes in the production line or in the workspace environment, robots can perpetually repeat the same movements to solve faster and more accurately a specific task in a pre-defined environment. In this collaborative setting, the human agent can solve parts of the task that are difficult for the robot to perform (e.g. personalized design [47]) and vice versa (e.g. manufacturing [47]) [45]. This innovative approach has been gaining traction in recent years, with the number of operational collaborative robots rapidly increasing (Fig. 1.2).

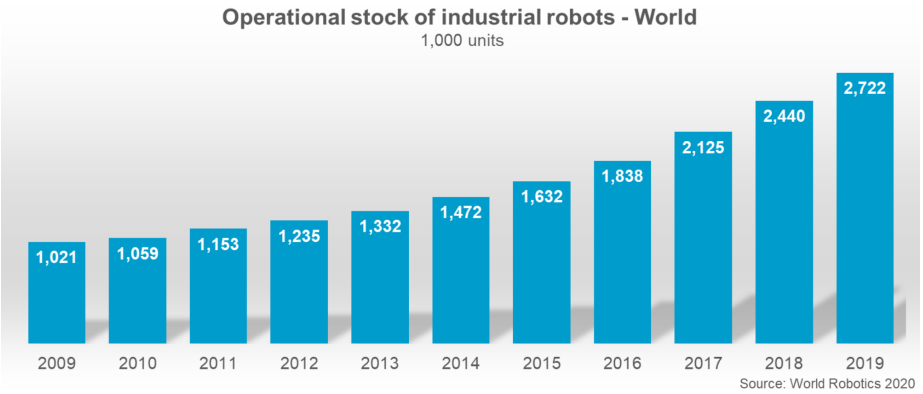

Figure 1.1: The global operational stock of industrial robots statistics [44].

As mentioned previously, the human and the cobot share the same workspace, which entails that these two agents are in close proximity to each other. On the other hand, traditional industrial robots are located in safety areas surrounded by fences away from

the workers in order to avoid work accidents [45]. These two different workspace environments prompt different concerns regarding safety issues; in the traditional industrial robots' setup, the workers' safety is guaranteed by the surrounding fences, while, when sharing the same workspace, the workers' safety must be guaranteed by the robot's behaviour and underlying programming.

Safety safeguards are a necessary prerequisite for robots and humans to share a common workspace. However, a different set of safety requirements is needed for the robot and the human to coexist and to collaborate. This is the case because *human-robot coexistence* and *human-robot collaboration* must be differentiated. In a human-robot coexistence scenario, the robot and the human agent share the same workspace and may even work with the same object. Nevertheless, the agents' actions are not coordinated and there is no contact between them [45]. On the other hand, in a human-robot collaboration scenario, the agents interact and coordinate with one another to achieve a common goal. In this case there may or may not be physical contact between the human and the robot [45].

In the HRC context, a robot's allowed movements are constrained to ensure the worker's safety. Safety concerns can be classified into two broad categories: *physical safety* and *psychological safety* concerns [46]. Physical safety is related to physical contact between the robot and the worker, while psychological safety is related to the discomfort and stress caused by the Human-Robot Interaction (HRI). When working within quasi-static environments, the robot can utilize a human-aware motion planning strategy and replan its motion whenever the actual movements do not follow the original plan. Nevertheless, following this approach in more dynamic environments is not feasible, since the motion plan would be frequently in need of revision and revising might not be possible in real-time for close proximity scenarios. For these environments, predicting the action of the human could prove crucial to guarantee the safe and efficient collaboration of the two agents, since the robot would be able to choose the appropriate action, the appropriate timing to start an action and/or the appropriate motions that should be followed to achieve an action. In this way, the robot would be able to avoid both physical contact (physical safety) and stressful near-collision trajectories (psychological safety) [46].

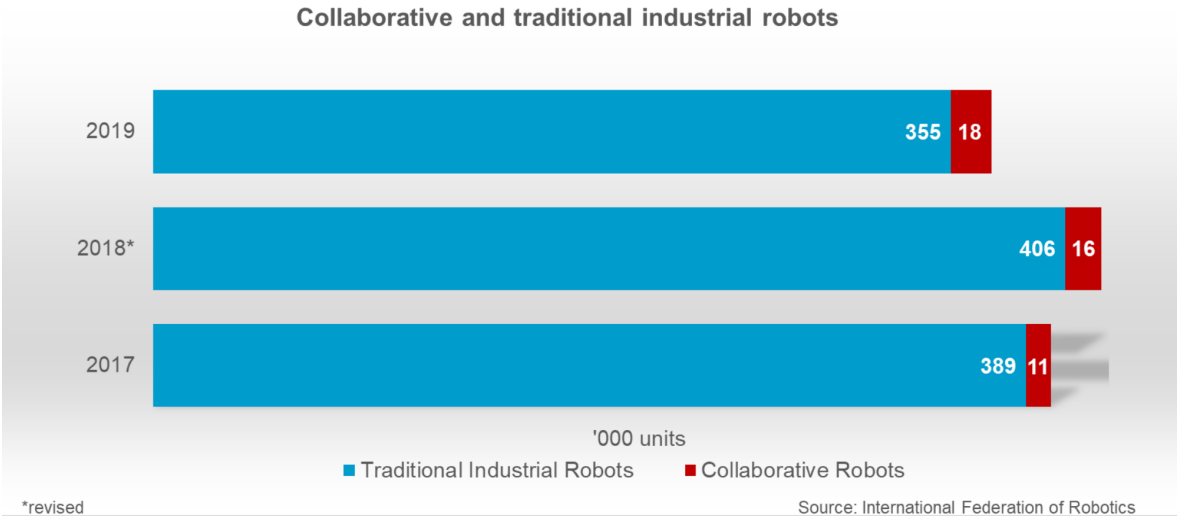

Figure 1.2: Comparison between the annual global sale volumes of traditional and collaborative industrial robots [44].

Apart from safety safeguards, human-robot coexistence and human-robot collaboration also require a different set of robotic capabilities. In order to identify the necessary capabilities for the latter case, one can first consider the qualities needed for effective human-human collaboration. Specifically, for two co-workers to be efficient in a common task, they must be able to exchange the roles of the passive and the active agent during their interaction. Similarly, in a HRC scenario the robot needs to perform both the active and the passive parts of the interaction [42]. In this way, the interaction between the human and the robot can be smooth, without long and noticeable pauses. To assume the active role during the interaction and to collaborate fluently with the human, the robot must be capable of understanding the action which is performed by the human at the time of the interaction [67].

In conclusion, predicting the action of the human agent can be crucial for HRC applications at multiple levels. In a production line setting, the workers' actions involve grasping different objects and, therefore, the different actions performed by the workers differ from each other only because of the object to be grasped. Therefore, predicting the object-to-be-grasped before the completion of the worker's movement would be important for a production line HRC application. In this line of work, we focus on the task of predicting the object towards which the human agent moves its hand based solely on the hand movement itself, without taking the object's properties (e.g. color, size) into consideration.

1.1. Motivation

Our work is based on the behavioral observation that the size of an object can be predicted by early information of hand kinematics [48]. Asuini and colleagues asked participants to reach and grasp two target objects of similar spherical shapes but different sizes: a hazelnut (small object) and a grapefruit (large object). In order to track the subjects' hand movements, each participant was outfitted with retro-reflective markers and a near-infrared motion capture system of 8 cameras with a frame rate of 100 FPS was used. The raw skeletal hand data was then used to extract hand kinematic features, such as the grip aperture (thumb tip - index tip distance), the wrist speed and the hand configuration (wrist - thumb tip - index tip configuration). After identifying the grasping phase of the movement using the wrist speed, the kinematic features were calculated at 10% intervals of the grasping movement. A classification analysis was performed for each of these time points using all the aforementioned features for the particular time point. The study's results showed that, even in the early stages of the movement, the target object could be predicted successfully above chance level. More specifically, the accuracy of the predictions during the 10% and 20% of the grasping movements was over 60% and 70% respectively, while the expected accuracy when guessing the target object at random would be 50%. Moreover, the accuracy of the predictions increased over time and after 60% of the movement the accuracy rate was approximately 100%.

These results illustrate that hand kinematics can be used to predict which object will be grasped with great success. Such findings could be explored in HRC applications, in order to enhance the workspace safety and the fluent collaboration between the human and the robot agents. Nevertheless, both the solution's cost, ease of installation, and its acceptance by the human agent should be first considered. Regarding the cost, the raw hand skeletal data in [48] was captured using a high-end motion capture system to extract the kinematic features with a minimal amount of noise. While the high temporal and spatial resolution of the motion capture set-up is helpful to identify the grasping

phase of the movements and, finally, to predict which object will be grasped, the price of a motion capture system could be prohibitive in many cases. At the same time, a percentage of the human workers could find using markers obtrusive, constraining natural human movements. Moreover, if this solution was adopted for HRC settings, the cameras of the motion capture system should be appropriately placed in the workspace introducing several installation requirements that might be prohibitive in many set-ups.

Therefore, *this work's goal is to investigate whether hand kinematics can be useful in predicting the object to be grasped when the movement is captured with a single RGB-D camera that is inexpensive, does not involve cumbersome installation and does not require outfitting the human agent with any markers.* In this setup, a hand pose estimation algorithm can be used to extract the skeletal data. However, as opposed to skeletal data recorded by a motion capture system, the keypoint detection by the state-of-the art hand pose estimation algorithms usually contains significantly more noise. As a result, the object prediction problem can be considered more challenging in this setup in comparison with the setup described in [48].

Besides using a single RGB-D camera, in a HRC scenario robot vision must be able to make real-time inferefences and to do so in a reasonable time horizon that makes sense both in terms of the human action (time to reach an object) and in terms of robot hardware capabilities (robot movement reaction times).

Considering all the above, the requirements of a HRC scenario where a robot must predict and act in time to safely and fluently collaborate with a human are summarized below:

- single RGB (-D) camera: The RGB and/or the depth information must be recorded for a movement using a single visual sensor; multiple-camera setups are not desired.

- real-time inference: In a real world human-robot collaboration scenario, predicting the human's action is time-sensitive and must occur in real-time. The required inference time depends on the available hardware, the Machine Learning (ML) model's size and floating point operations (FLOPs) and how hardware friendly the model's architecture is [49].

- early prediction: The target object must be predicted early enough for the robotic arm to decide and act in a way that supports safe and fluent collaboration.

1.2. Our work

As already stated, this work is focused on the task of predicting which object will be grasped based exclusively on the hand movement (without considering the object's properties) through information provided by a single RGB-D camera. To this end, the dataset used for this work was collected by conducting a similar experiment as in [48], but without the use of a high-end motion capture system and the use of markers. Instead, a single RGB-D camera was used to record the movements [51].

Since the dataset consisted of RGB-D video data, a hand pose estimation algorithm had to be applied to the data of each recorded movement to extract the hand's skeletal data. Although the thesis did not focus on this task, we reviewed various state-of-the art

methods (Chapter 2) that could be used for hand pose estimation. For final selection we required that the method:

a) supported the thesis' second requirement for real time inference,
b) was open, and
c) was integrated in the Robot Operating System (ROS)[1] [59].

Furthermore, in Chapter 2 we also reviewed good practices for time series classification and state-of-the-art approaches for the related task of human-action recognition.

In Chapter 3, we describe the movements of the dataset and the extraction of the hand skeletal data. Furthermore, the steps of the preprocessing pipeline which were applied to the skeletal data of the movements and the features that were engineered for the grasping movements are analyzed. After preprocessing and feature engineering, both "traditional" ML models and a baseline DL convolutional neural network model were considered. The methodologies which were followed for both of these two cases are also described in this chapter. Finally, at the end of the chapter, the evaluation strategies which were used in order to assess the models are introduced.

The models are evaluated and compared to each other in Chapter 4 for different grasping movement completion percentages, kinematic features and dataset split strategies, as described in the evaluation strategies of the previous chapter. Furthermore, the prediction accuracies of the top-performing models are evaluated with respect to robot capabilities based on real-time robot data. Finally, in Chapter 5 we arrive at our conclusions based on the models' evaluation results and propose some promising future research directions.

[1] https://www.ros.org/

2. LITERATURE REVIEW

As previously mentioned, the movements of the dataset which was used for this work were captured with an *RGB-D camera*. At the same time, the goal of this thesis was to predict the object-to-be-grasped based only on the kinematic information of the *hand movement*. As a result, a *hand pose estimation* model must be used in order to extract the *hand skeletal data*. Given that in [48] the 3D keypoint coordinates were used, multiple **3D hand pose estimation** models were reviewed. The 3D hand pose estimation methods which were explored are presented in Section 2.1.

In Section 2.2, we explore state-of-the-art approaches for tasks which are closely related to ours. We focused on similar tasks, since, besides the behavioral paper [48], we did not come across other papers regarding the **object-to-be-grasped predictions**. Specifically, the task of predicting the object-to-be-grasped was considered to fall into the area of the more general **human action prediction** problem; temporally incomplete video data is given as input and the objective is to predict which human action would be performed next [53]. In this case, the model infers the action label before fully observing the entire action execution. While this problem was similar to our task, to the best of our knowledge previous approaches were based mainly on the *RGB input modality* instead of skeletal data [61-66]. Since our work is based on hand skeletal data, we did not investigate any further the state-of-the-art human action prediction approaches.

Following that, a similar category of problems was considered. Specifically, we considered the **human action recognition** problem, where a video containing a complete action execution is given as input and the goal is to recognize which is the action [53]. In this problem, the model infers the action label after the entire action execution is observed. Previous approaches for the action recognition task were based either on RGB, depth or skeletal data input modalities or on a combination of them. The various categorizations of the human action recognition approaches are illustrated in Fig. 2.1. Apart from the input modality, the human action recognition approaches can be categorized as either *offline* or *online*. An *offline* model receives as input a segmented video sequence, which is labeled with a single action label for the entire sequence. Even though *continuous action recognition* is one improved version where the videos are untrimmed, it still assumes that the whole video is available before processing. Differently from continuous action recognition, *online* action recognition aims to receive continuous streams of unprocessed visual data and recognize actions from an unsegmented stream of data in a continuous manner. Given that for our task the skeleton-data are used as the input modality and that each video of the dataset contained only one grasping movement, we explored thoroughly the state-of-the-art approaches for **offline skeleton-based human action recognition**.

Last but not least, following the application of a hand pose estimation algorithm, the skeletal data would be extracted for all the frames of a movement. Therefore, the skeletal data of a movement could be regarded as a *multivariate time-series*. In this case, the variables of the time-series would be the hand joints' coordinates. Moreover, the same would apply if for each frame of the movement we engineered kinematic features (similar to [48]) based on the skeletal data. In this case, the data of a movement could be regarded as a *multivariate time-series* with the kinematic features considered as the time-series' variables. Consequently, in Section 2.3 we reviewed good practices regarding **time-series classification**.

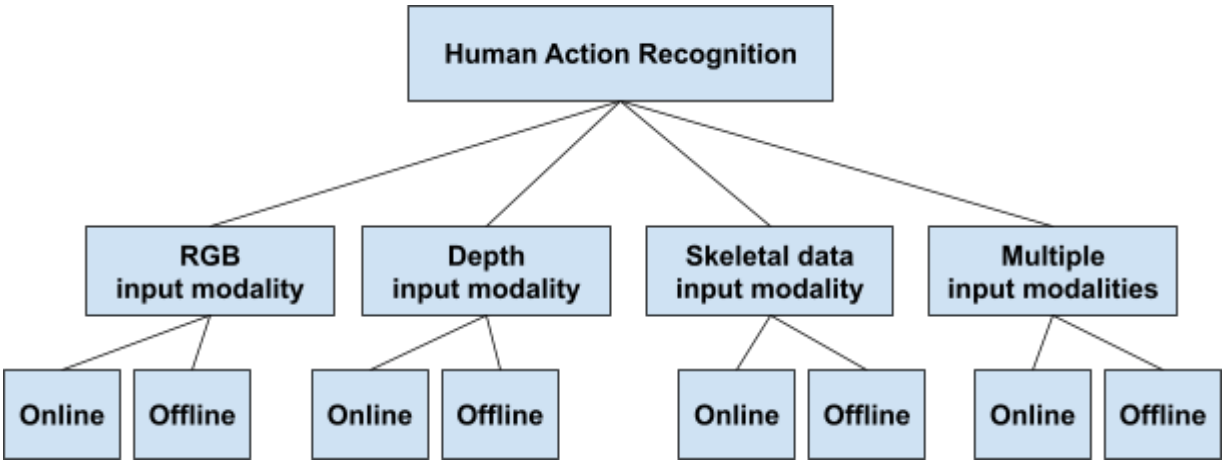

Figure 2.1: A categorization of the human action recognition approaches.

2.1. 3D Hand Pose Estimation

The 3D Hand Pose Estimation task is considerably difficult due to varying hand shapes, viewpoints, joint visibility and articulation distributions. There are two main categories of methods based on the input modality: a) *Depth-based* and b) *RGB-based* 3D Hand Pose Estimation methods. The depth-based methods are presented in subsection 2.1.1 and the RGB-based methods in subsection 2.1.2. For each of the 3D Hand Pose Estimation methods which were reviewed we evaluated its *real-time inference ability*, whether it was open-source and whether it was *ROS-integrated*.

2.1.1. Depth-based 3D Hand Pose Estimation

Depth-based methods are the main methods that are traditionally used for 3D Hand Pose estimation [26]. This is because depth maps encode 2.5D information, which means that, while the depth map is still a 2D grid, it also provides information for a third, depth dimension. The 2.5D depth information of a depth map can also be represented as a point cloud or 3D voxels. Different representations of the same information may lead to learning features of the hand region more effectively. The depth-based approaches can generally be divided into detection-based [30,31] and regression-based [32-36] methods based on the network's output. Detection-based methods produce dense representations (e.g. probability density heatmaps, offset or vector fields) for each joint and the 3D joint positions are estimated with a non-learnable operation (e.g. argmax) guided by the dense representations during post-processing. These methods are better at capturing local information, since a spatial layout is preserved through convolutional layers. Regression-based methods directly output the 3D joint coordinates and are better at capturing global information, since the features of different regions are aggregated in fully connected layers.

Earlier approaches employed traditional machine learning methods to tackle this task. For example, Microsoft Kinect captures a depth image and afterwards uses a Random Forest model to classify each pixel as a part of the hand or the body [28]. Supancic et al. [29] use a nearest neighbour approach with volumetric exemplars, which outperformed most previous learning-based approaches. Moreover, [29] concluded that deep learning methods generalize better to novel hand poses compared to other

learning-based methods, even without using large training sets. In accordance with these results, nowadays state-of-the-art approaches involve deep learning methods.

Wan et al. [30] propose a detection-based method with dense pixel-wise estimation in order to leverage both the 2D and 3D properties of the depth map input. In this approach, a 2D CNN network outputs 2D projection heatmaps, 3D closeness heatmaps and dense 3D unit vectors. The 3D closeness heatmap and the 3D unit vector make the estimate translation-invariant and increase the generalization to different combinations of finger poses. The 2D projection heatmap enforces consistency between the 2D projections of the 3D joints estimations and the pixel-wise 2D joint detections. The 3D joint coordinates are calculated during post-processing based on a variance of the mean shift algorithm.

Ge et al. [31] propose to exploit a point cloud representation of the information. The depth image is first converted to a point cloud using the camera's intrinsic parameters. Following that, the 3D points of the point cloud are transformed into the coordinate system of an oriented bounding box and they are normalized before point sampling. A closeness heatmap and a 3D unit vector field are produced for each hand joint. The sampled, normalized points are fed into a stacked PointNet to infer a closeness value and a 3D offset unit vector for each sample point - joint pair. The network is optimized based on the produced closeness heatmaps and unit vector fields. During inference, the 3D joint coordinates are estimated as a post-processing step.

The hand pose, hand shape and camera viewpoints can be regarded as independent variables in the hand pose estimation task. Due to the high dimensionality of the problem, a training set that covers the entire domain of the problem does not exist. Moreover, generalizing to unseen articulations, viewpoints and shapes results in an accuracy drop. A possible solution to remedy this problem, is to leverage synthetic data in order to fill the gaps of existing training sets, since a synthetic training set can be virtually infinite [27]. Following this approach, Rad et al. [32] use synthetic data for training and also focus on the issue that the distribution of synthetic depth maps' features is not the same as the distribution of real depth maps' features. Since these two distributions are not identical, using synthetic data without a domain adaptation technique can result in suboptimal performance. The proposed feature mapping network can be decomposed into 3 basic components: a feature extractor that outputs a feature vector when given an image, a 3D pose predictor which outputs the joints' locations when given a feature vector and a mapping network that outputs a feature vector following the synthetic images' distribution when given a feature vector that follows the real images' distribution. The feature extractor is trained both with real and synthetic images. The pose predictor is trained with domain adaptations of real images' features and with synthetic images' features. Finally, the mapping network is trained using paired real and synthetic images' feature vectors. The real and the synthetic image of a pair have the same pose and the mapping network's goal is to minimize the distance between the feature vector of the real image and the output feature vector of the mapping network. For each pair, only a few feature coefficients are responsible for the domain gap, and they can be attenuated by the mapping. During inference, given a real depth map of a hand, the image features are first computed (feature extractor), then they are mapped to the feature space of synthetic images (mapping network), and finally the resulting features are used as input to predict the 3D hand pose (3D pose predictor).

Similar to [32], Zhang et al. [33] synthesize training data and propose a data synthesis strategy, which combines real and synthetic images to form a training set. This method

is based on MANO [54], a model which renders a depth image of the right hand, based on a camera parameter, a hand pose parameter and a hand shape parameter. The MANO parameters of real images are calculated with gradient-based optimization. The final training dataset contains a Real dataset with the depth images of the original dataset, a Synthetic dataset with the synthetic images rendered by MANO with the MANO parameters of the real images, a Mixed Synthetic dataset where real and synthetic images are combined with a linear blending method and, finally, a Noised Synthetic dataset where Gaussian noise is applied to the MANO parameters of real depth images and then MANO renders the synthetic images based on the noised parameters. The proposed network consists of 2 stages, both of which use a 2D CNN as backbone. First, a coarse patch, previously extracted by the input depth image, is fed into the first stage of the network to estimate an initial hand pose. Then, this initial hand pose is used to extract an augmented hand region from the coarse patch with a 3D bounding box, in order to remove noisy regions. Finally, this augmented hand region is fed into the second neural network to estimate the final hand pose.

In another approach to utilize synthetic data, Baek et al. [34] synthesize skeleton data instead of depth maps, as opposed to [32] and [33], and leverage both real depth map and skeletons pairs and synthetic unpaired skeletons. The synthetic skeletons are constructed by applying 3D rotations to cover more viewpoints and by changing the hand shape. The proposed network consists of a hand pose generator (HPG), a hand pose estimator (HPE), a depth hand pose discriminator (HPDx) and skeleton hand pose discriminator (HPDy). The hand pose generator, which follows the conditional Generative Adversarial Network (GAN) architecture, synthesizes a depth map when given a skeleton, while the hand pose estimator generates the 3D joint coordinates based on a depth map input. The depth hand pose discriminator is a standard GAN discriminator, which distinguishes real from synthetic depth maps and, finally, the skeleton hand pose estimator is used to distinguish whether the finger joints fit the human skeleton model. During training, the network is optimized using the paired real data to minimize the error of HPE and to enforce the cyclic consistency for both HPE-HPG and HPG-HPE. Moreover the network is optimized using the unpaired synthetic data to enforce HPG-HPE-HPG consistency. During inference, when given a depth map the HPE outputs an initial pose estimation, which is further refined with gradient backpropagation from HPG and HPDy.

The main reason why existing datasets are not large enough to be representative of the whole domain is that 3D Hand Pose annotation is not an easy task. Apart from synthesizing data, semi-supervised learning can also be applied to close the gap between the training set and the problem domain by leveraging unannotated data. Chen et al. [35] propose an encoder-decoder network architecture for semi-supervised learning. The input depth image is first converted to a pointcloud, which is then sampled and normalized, similar to [31]. The encoder maps hierarchically the pointcloud to point features, node features and a global feature vector. During training with unlabeled data, the global feature vector is fed to the decoder in order to reconstruct the original pointcloud, which will result in a better representation of the global feature and will help the encoder to capture more informative features. During training with labeled data, all the features of the encoder are also concatenated to a single vector, which is then fed to a hand pose regression module to output the 3D joint coordinates.

Method-wise ensembles, where multiple predictions are fused together to obtain the final prediction, can improve the performance of a model. To this end, Xiong et al. [36] propose an end-to-end deep learning model which utilizes ensemble learning. Specifically, the model predicts the 3D joint positions by aggregating the estimated

offsets of multiple anchor points. In this regression-based method, the anchor points are densely set on the input depth image and all of them contribute to the final predictions, but with different weights for each joint. In particular, the informative anchor points of a joint (the ones that are assigned a greater weighting factor) are encouraged to be uniformly close to it, since both the estimations of the anchor points and the location of informative anchor points are taken into consideration by the training loss function. Each anchor point makes two separate predictions for a joint, one prediction to estimate the in-plane offset between the joint and the anchor point and a second prediction to estimate the depth offset between the joint and the center of the hand. In order to achieve real-time performance, the backbone of the model is a 2D CNN without any time-consuming 3D convolutional and deconvolutional layers and the input modality is a depth map without any complex preprocessing procedures, such as voxelizing or point sampling.

Huang et al. [37] propose a hybrid method which takes advantage of both detection- and regression-based methods' strengths. In this approach, the network first extracts a dense representation for each joint as output of the backbone network and afterwards the predictions of different regions are adaptively aggregated to regress the joint coordinates, in a form of ensemble learning similar to [36]. This pipeline allows for both dense and joint supervision. The backbone network is a 2D CNN with deconvolution layers that are used to lift the resolution of the feature map to the resolution of the dense representations. The dense representation type is 3D offsets, which can be further decomposed into 3D directional unit vectors and closeness heatmaps. Since these two dense representation types are independent and have different physical characteristics, the backbone network uses two separate convolutional heads to generate them.

The aforementioned depth-based 3D hand pose estimation models were evaluated with respect to common datasets. Specifically, the MSRA15 [57], NYU [55] and ICVL [56] 3D hand pose estimation datasets and the datasets of the challenges Hands In a Million 2017 (Hands 2017) [21] and Hands In a Million 2019 (Hands 2019) [27] were often used to evaluate the performance of a model. The primary metric the models were evaluated with was the average 3D distance error. The results of the models are presented collectively in Table 2.1 with the average errors expressed in millimeters (mm). The minimum error is highlighted for each dataset. Moreover, for each model it is cited whether it is open-source and ROS-integrated. Finally, the inference frequency of the models is presented. The inference frequency of a model expresses the number of frames that can be fed to the model in one second for inference. The inference frequency is dependent on the available hardware. Therefore, the GPU which was used to measure the model's frequency is also mentioned. Based on the table, the accuracy of the pose estimation algorithms was impressive for all the models and, particularly, for the Adaptive Weighting Regression (AWR) model. Moreover, there were multiple models with large enough inference frequencies for real-time inference and the majority of the hand pose estimation models were open-source. On the other hand, none of the listed models was integrated in the Robot Operating System (ROS).

Table 2.1: The evaluation of depth-based 3D hand pose estimation models

Model	Average 3D Distance Error (mm)					Inference Frequency	Open Source	ROS integration
	MSRA 15 [57]	NYU [55]	ICVL [56]	Hands 2017 [21]	Hands 2019 [27]			
A2J [36]	-	8.61	6.46	8.57	13.77	105.06 NVIDIA GTX 1080 Ti	✔	✘
Feature Mapping [32]	-	**7.44**	-	-	-	116.27 NVIDIA GTX TITAN X	✘	✘
AWR [37]	**7.20**	7.48	**5.98**	**7.48**	-	-	✔	✘
Hand Augment [33]	-	9.02	-	-	**13.66**	**-**	✔	✘
Dense Regression [30]	7.23	10.21	7.24	-	-	27.8 NVIDIA GTX TITAN X	✔	✘
P2P [31]	7.71	9.04	6.32	-	-	41.8 NVIDIA TITAN Xp	✔	✘
SO-Hand Net [35]	-	11.2	7.7	24.65	-	58 NVIDIA TITAN Xp	✔	✘
Augmented Skeleton Space Transfer [34]	12.5	14.1	8.5	-	-	300 2 NVIDIA GTX 1070	✔	✘

2.1.2. RGB-based 3D Hand Pose Estimation

While depth images are able to capture 2.5D information of the world scene, RGB images only capture 2D information. Another drawback of RGB images, compared to depth maps, is that two images with the same hand shape, viewpoint, joint visibility and articulation distribution could still vary due to different lighting conditions and colors [27]. Reducing the dimensionality of the input modality makes the task significantly harder. However, on the upside, an RGB-based method has increased generalization power, since the vast majority of existing visual data is RGB-only (e.g. YouTube) [26].

Similar to the depth-based 3D Hand Pose Estimation task, annotating a dataset of RGB images with 3D joint coordinates typically requires calibrated multi-view setups or laborious manual annotations. Nevertheless, real RGB-D unlabeled data can be easily obtained with a cheap RGB-D camera and synthetic labeled RGB-D data are easy to generate. Cai et al. [38] propose to leverage the depth maps as a weak label both for the unlabeled real data and for the labeled synthetic data. The network consists of three main components: a 2D pose estimation network, a 3D regression network and a depth regularizer. The 2D pose estimation network follows an encoder-decoder architecture that outputs 2D heatmaps when given a cropped RGB image containing a hand. The network, also, outputs intermediate features, which are then concatenated with the 2D heatmaps to be fed into the 3D regression network. This network outputs the 3D hand joint coordinates. The depth regularizer outputs a depth map when given a 3D hand pose. During training, real and synthetic images are mixed. When given a real image, the network is optimized based on the output of the 2D pose estimation network and on the output of the depth regularizer, which is used to employ weak supervision on the dataset. When given a synthetic image, the network is also optimized based on the output of the 3D regression network, since the 3D coordinate labels are available. During inference, the depth regularizer component is not used.

While unlabeled depth maps are easy to acquire, the vast majority of image data is RGB without any available depth information. To this end, Mueller et al. [39] propose leveraging synthetic data to train a network based on RGB-only data. Since the distribution of synthetic RGB images does not match the distribution of real RGB images, a domain adaptation technique is used in order to map a real RGB image to the synthetic image domain. More specifically a translation GAN is trained to enforce cyclic consistency and geometric consistency, which preserves the hand pose before and after translation. In order to regress the 2D and 3D joint predictions, a CNN is trained on synthetic data. The 2D predictions are represented as 2D heatmaps, while the 3D predictions are relative to the root joint. These predictions are complementary to each other, as heatmaps represent uncertainties and 3D predictions resolve depth ambiguities. After the 2D and 3D predictions are extracted from the regression CNN, they are refined using a kinematic skeleton model to ensure anatomically plausible hand pose.

For the task of hand pose estimation, 2D annotations are easier to acquire compared to 3D annotations. However, weak supervision with 2D joint coordinate labels usually improves performance for in-plane detection, but it doesn’t resolve the depth ambiguities. To solve this problem, Spurr et al. [40] propose to enforce biomechanical constraints in order for the predicted 3D hand pose to be anatomically feasible. These constraints are enforced on bone lengths, palm structure and joint angles via the loss function during training. The network is trained using a dataset with 3D annotations and a dataset with 2D annotations. When only weak 2D annotations are available, the network is optimized based on the projections of the 3D predictions, as opposed to when 3D annotations are available. In this case the network is optimized based on the discrepancies between the predicted and ground-truth 3D coordinates.

Last but not least, Cao et al. [50] introduce a real-time multi-person framework, OpenPose, for 2D Pose Estimation. Apart from hand keypoints, the framework can be used to detect keypoints of the body, foot and/or face. The proposed approach is based on a nonparametric Part Affinity Fields (PAFs) representation. This representation allows the model to learn to relate body parts with individuals in the image and, because of it, the inference frequency of the model is independent of the number of people in the image. Furthermore, in case multiple synchronized camera views are available, the

framework can perform 3D keypoint detection after the 2D results of all the camera views are triangulated and refined. However, the videos of the dataset which we examined for this work were recorded with a single RGB-D camera. A simple approach which can be followed in order to lift the 2D OpenPose keypoint detections to the 3D space for a single RGB camera is to leverage the depth map information after the OpenPose inference [60]. Specifically, the 2D coordinates of a keypoint correspond to a pixel of the depth map and the value of this pixel is assigned as the depth of the keypoint. Following that, the 3D keypoint coordinates can be calculated based on the 2D keypoint coordinates, the keypoint's depth and the camera's intrinsic parameters. A downside of this approach is that in case of occlusions or self-occlusions the 3D detection would not be accurate, since the actual depth of the keypoint would not be equal to its assigned depth.

Table 2.2: The evaluation of RGB-based 3D hand pose estimation models

Model	Inference Frequency	Open Source	ROS integration
Cai et al. [38]	-	✔	✘
Mueller et al. [39]	-	✔	✘
Spurr et al. [40]	-	✘	✘
OpenPose [50] + depth map	22.0 NVIDIA GTX 1080 Ti	✔	✔

The aforementioned RGB-based 3D hand pose estimation methods are compared in Table 2.2. Similar to Table 2.1, for each model it is cited whether it is open-source and ROS-integrated. Moreover, the inference frequency of the models is presented with respect to the GPU which was used to measure it. Since these approaches were not evaluated on common datasets, a comparison of the detections' accuracies is not possible. The majority of the methods were open-source, but only the approach which combined OpenPose with the depth map information was ROS integrated. Similarly the inference frequency is listed only for the OpenPose-based approach. This frequency is probably low for real-time inference in case one wants to extract the skeletal data for all the frames using the listed GPU.

2.2. Offline Skeleton-based Action Recognition

The conventional input modalities for the action recognition task are RGB image sequences, depth image sequences or a fusion of these modalities. Compared with skeleton data, these conventional modalities are more computation and time consuming and less robust when facing complicated background or changes in body scales, viewpoints and motion speeds [1]. The vast majority of the skeleton-based action recognition literature focuses on the skeletal data of the full body, instead of the hand. Following that, in order to investigate the state-of-the-art methods for this task the approaches which were reviewed use the skeletal data of the full body. In this context,

the models are tasked to recognize actions such as “sitting down”, “drinking water”, “reading”, “pointing to something with finger”, etc. Therefore, the models that are presented in the following subsections are not tailor-made for a limited set of similar actions. Instead they are evaluated on datasets, such as NTU-RGB+D [18], that contain a wide range of actions. Skeleton-based action recognition approaches can be divided into two broad categories based on their methodology: **hand-crafted features** approaches and **deep learning** approaches. The former are presented in subsection 2.2.1 and the latter in subsection 2.2.2.

2.2.1. Hand-crafted features Approaches

In the traditional **hand-crafted features** approaches, there are two sequential stages, a feature extraction stage followed by a feature representation stage, to build the final descriptor. First, during the feature extraction stage, carefully chosen features are extracted from a skeleton sequence. Then, during the feature representation stage, these extracted features are concatenated, fused together or even fed into a machine learning model in order to obtain the final feature descriptor [2]. This descriptor represents the motion patterns of the skeleton sequence and is eventually fed into a machine learning model to train the model and/or infer the action label.

Vemulapalli et al. [3] model the 3D geometric relationships between body parts using 3D rotations and translations in a Lie group. Following that, the curves in this Lie group are mapped to its Lie algebra vector space. The Lie algebra vectors are then classified using a combination of dynamic time warping, Fourier temporal pyramid representation and linear Support Vector Machine (SVM).

Koniusz et al. [5] propose to use tensor representations in order to capture the higher-order relationships between 3D human body joints for action recognition. In this context, two different Radial Basis Function (RBF) kernels are applied, one to capture the spatio-temporal compatibility of joints and one for the action dynamics of a sequence. Subsequently, these kernels are linearized to build a tensor representation and a Support Vector Machine model is trained with these tensor representations.

2.2.2. Deep Learning Approaches

As opposed to hand-crafted features approaches, **deep learning** methods are data-driven. As a result, feature engineering is not that important, while having a large volume of labeled data is. Deep learning skeleton-based approaches can be further categorized based on the neural network’s architecture as:

- **Recurrent Neural Net- (RNN-) based approaches**
- **Convolutional Neural Net- (CNN-) based approaches**
- **Graph Convolutional Neural Net- (GCN-) based approaches**
- **Combined approaches**

2.2.2.1. Recurrent Neural Net- (RNN-) based approaches

An RNN unit receives as input on the current timestep t the output of the unit on the previous timestep t-1. This recursive connection has proven to be effective when dealing with sequential data. However, a skeleton sequence, apart from the temporal-sequential information, also contains spatial information in each frame.

Due to the weakness of conventional RNNs in spatial modeling, Wang et al. [6] propose a two-stream RNN architecture with a temporal and a spatial stream. The input of the temporal RNN stream at each step is a vector with the 3D joints' coordinates for a single frame concatenated, while the input of the spatial RNN stream is a sequence of joints. The scores of the two streams are then fused together in order to output the final prediction scores.

To address the exploding and vanishing gradient issues and the long-term temporal modeling problems of the standard RNN, Gated Recurrent Units (GRUs) or Long Short-Term Memory (LSTM) units are often used instead of RNN units. Moreover, an attention mechanism has proven to be extremely effective when used in conjunction with RNN architectures, in order to focus on more discriminative features. For example, in the action recognition task, Li et al. [7] use a recurrent relational network to learn the spatial features, which is followed by a multi-layer LSTM to learn the temporal features in skeleton sequences. An adaptive attention module is also used to focus on discriminative joints towards a certain action. In another example, Song et al. [8] embed a spatiotemporal attention module on top of a LSTM recurrent network, in order to pay more attention both to the joints and the frames which are characteristic of the action that is performed.

2.2.2.2. Convolutional Neural Net- (CNN-) based approaches

While RNN models focus on sequential data tasks, CNN models generally focus on image-based tasks. In order to use CNNs for the action recognition task, the skeleton sequence data are first transformed from a vector sequence to a pseudo-image based on manually-designed transformation rules. In some approaches, instead of representing both the spatial and temporal information as a single pseudo-image, multiple pseudo-images are used and then fed into the CNN.

For example, Li et al. [11] first divide the joints of each frame in five parts based on natural body structure and then apply a translation-scale invariant mapping strategy and data augmentation techniques to these parts to obtain a 2D image, which is afterwards fed to a pretrained CNN. Li et al. [15] propose a shape-motion representation from geometric algebra, which is robust against view variations and utilizes both joints' and bones' motion and shape information.

Most CNN-based approaches encode temporal and spatial information as rows and columns respectively. However, with this type of representation, the information of joints in non-neighboring columns will not be taken into account to learn a co-occurrence feature. For example, if the keypoints of the right and the left hand are placed in non-neighboring columns, a CNN-based model might not be able to learn a co-occurrence feature for the hands and, as a result, it would be difficult to identify actions such as "clapping". Li et al. [13] address this issue with a hierarchical methodology, where the information of each joint is first independently encoded and then assembled into spatial and temporal high-level representations. To alleviate the same concerns, Cao et al. [14] use a fully convolutional permutation network to optimize the order of the joints. Liang et al. [16] propose a three-stage network with a joint, a bone and a motion stream, where the first-stage network learns co-occurrence features from the features of each stream, the second-stage network fuses the co-occurrence features of different streams in a pairwise manner and, finally, the third-stage network uses multi-task or ensemble learning for training and inference respectively.

2.3.2.3. Graph Convolutional Neural Net- (GCN-) based approaches

One of the main concerns both for RNN- and CNN-based approaches is that the skeleton structure cannot be effectively represented with a sequence vector or a pseudo-image respectively. On the contrary, the human skeleton is naturally structured as a graph in a non-Euclidean space with the joints as vertices and the bones as edges. To this end, GCN models are used in the skeleton-based action recognition task to fully exploit the graph structure of the skeleton data and to fully express the dependency between correlated joints.

As a first attempt, Yan et al. [17] propose a spatial-temporal graph convolutional network where the joints of each frame are represented as vertices, the bones are represented as spatial edges and the corresponding joints in two consecutive frames are connected with temporal edges. In this work, the sampling area of the graph convolution is defined as the adjacent (within distance 1) vertices and is divided into three subsets, the target vertex itself, the subset of the adjacent vertices closer to the center of gravity than the target vertex, and the subset of the adjacent vertices further from the center of gravity than the target vertex.

In this first attempt, the GCN is based on a fixed skeleton graph that captures only the dependencies between physically nearby joints. A negative effect of this graph construction method is that co-occurrence features of non-local joints are not captured and that the same fixed graph topology may not be optimal for different examples. Shi et al. [19] address these issues and also focus on the hierarchical structure of GCNs. Different layers of a GCN contain different semantic information. As a result, a fixed graph topology over all the layers of the GCN cannot fully exploit the semantic information extracted in different layers. For this purpose, the model learns the topology of a global and an individual graph in each layer of the GCN. The global graph topology is optimized for the entire training dataset, while the individual graph topology is optimized based on the activations of each instance of the dataset. Finally, the adjacency matrix in each layer is defined as a linear combination of the normalized natural connectivity adjacency matrix (defined in [17]), the global adjacency matrix and the individual adjacency matrix. Moreover, apart from joint information, the model makes use of the bones' information. The scores of the joint and bone streams are fused together to obtain the final classification scores. In a later approach, Shi et al. [20] build on their previous results [19] and extend their previous model with more streams of information and an attention mechanism. A Spatial-Temporal-Channel (STC) attention module with three sequentially arranged attention submodules is used. This module is incorporated in the beginning of every graph convolutional block, in order to adaptively recalibrate the inputs of each layer for each sample. In this way the model can focus more on discriminative joints, frames and channels. Moreover, the model consists of four identical streams, a joint stream, a bone stream and their corresponding motion streams, the scores of which are fused together similar to [19]. Moreover, in [20] the authors propose to fuse the results of the skeleton-based model with an RGB-based model following a pose-guided cropping strategy, in order to remove background noise from the RGB input modality.

The previous approaches focus mainly on maximizing the models' performance without taking into serious consideration the computational complexity. On the contrary, Cheng et al. [24] achieve state-of-the-art accuracy, while, simultaneously, minimizing the FLOPs of the model. To this end, a spatial and temporal shift graph convolution operation are used instead of the conventional spatial and temporal graph convolutions. Same as with the CNN shift convolution [4], the shift operation does not introduce extra

parameters or FLOPs to the model and, afterwards, a lightweight pointwise convolution follows. For the spatial graph convolution operation, a non-local strategy is used. In this strategy, the information of each channel is permuted across all the nodes of the graph, without any information loss and the receptive field of each node is the entire graph. A learnable mask is used to highlight the importance of each feature. The mask is optimized based on the entire dataset. Moreover, the authors acknowledge the deficiency of a single kernel of fixed size for the temporal convolution. They also point out that different layers and different datasets may need different temporal receptive fields. Therefore, even a fusion of kernels of different sizes or dilation rates is not optimal. For this reason, all the activations of a channel are shifted by the same temporal distance to another frame. The temporal shift distance of a channel is learnable and it is optimized globally on the whole dataset for each layer. The final model uses four streams using the same strategy as [20].

Cheng et al. [25], also propose an alternative efficient approach without introducing additional computational complexity. In the conventional graph convolution operation, the adjacency matrix is multiplied with the input feature matrix and, as a result, each channel is multiplied with the same adjacency matrix. Consequently, the aggregation of the spatial characteristics of the channels is not optimal, as different channels have different characteristics. Instead of following this conventional convolution operation, a decoupling graph convolution, DeCoupling GCN, is introduced. In this novel operation, the channels are split into groups and for each group a different trainable adjacency matrix is multiplied with the group's channels. The adjacency matrices are optimized globally on the entire dataset. This operation is orthogonal with multi-partition and the proposed model combines them. DeCoupling GCN increases slightly the parameters of the model by 5-10% due to the extra adjacency matrices. Nevertheless the computational complexity remains the same. Moreover, this approach tackles the problem of overfitting. In order to alleviate overfitting in Deep Neural Networks, a popular regularization technique is Dropout. However, this technique is not effective for GCNs, because, even if a node is dropped, information about the node can still be propagated from its neighbouring nodes. For this reason, an Adaptive DropGraph regularization technique is proposed, where each node may be discarded with some probability and for each discarded root node all the nodes that are within k steps are also dropped. An attention map is used to drop important nodes more often. DropGraph is applied sequentially on the spatial and the temporal dimension. In the spatial dimension DropGraph is applied based on the adjacency matrices of DeCoupling GCN, while in the temporal dimension it is applied to a graph of frames, where consecutive frames are connected with an edge. As in their previous approach [24], the final model uses four streams.

2.2.2.4. Combined approaches

In these combined approaches, the models consist of modules which have different architectures. The motivation behind this design is that different neural networks architectures can complement each other in order to achieve superior performance.

For example, Xie et al. [12] use a model with a Temporal Attention Recalibration Module and a Spatio-Temporal Convolution Module. The Temporal Attention Recalibration Module pays attention to potentially discriminative frames of the skeleton sequence using an RNN model. In the Spatio-Temporal Convolution Module, the attention calibrated skeleton sequences are then fed into a CNN to model further spatial and temporal information.

Zhang et al. [9] focus on the large viewpoint variance problem, which is generally encountered on the action recognition task. To this end, they propose a view-adaptive LSTM recurrent network and a view-adaptive CNN network that transform the skeleton data to more advantageous viewpoints, while they also preserve the continuity of the action. A data augmentation strategy is employed in order to increase the viewpoint diversity during training. The scores of the RNN-based and the CNN-based models are fused together to obtain the final scores.

Finally, Si et al. [10] propose a novel model which consists of a spatial reasoning network and a temporal stack learning network. The spatial reasoning network uses a residual Graph Neural Network to learn high-level spatial features, while the temporal stack learning network uses multiple skip-clip LSTMs to capture high-level temporal features. In this approach, the skeletal data are first fed to the spatial reasoning network, whose output is then fed to the temporal stack learning network in order to infer the action label.

Table 2.3: The evaluation of skeleton-based action recognition deep learning models.

Model		NTU-RGB+D [18]	
		CV (%)	CS (%)
RNN-based Approaches	Two-Stream RNN [6]	79.5	71.3
	ARRN-LSTM [7]	88.8	80.7
	STA-LSTM [8]	81.2	73.4
	VA-RNN [9]	88.9	79.8
CNN-based Approaches	3scale ResNet152 [11]	92.3	85.0
	FO-GASTM [15]	90.05	82.83
	HCN [13]	91.1	86.5
	(P+C)Net [14]	93.5	86.1
	3SCNN [16]	93.7	88.6
	VA-CNN [9]	94.3	88.7
GCN-based Approaches	ST-GCN [17]	88.3	81.5
	2s-AGCN [19]	95.1	88.5
	MS-AAGCN [20]	96.2	90.0
	4s Shift-GCN [24]	96.5	90.7
	DC-GCN+ADG [25]	**96.6**	**90.8**
Combined Approaches	MANs [12]	93.22	82.67
	VA-fusion [9]	95.0	89.4
	SR-TSL [10]	92.4	84.8

All the Deep Learning approaches which were previously presented were evaluated for the NTU-RGB+D [18] human action recognition dataset. The dataset consists of 60 actions, including the previously mentioned “sitting down”, “drinking water”, “reading”, “pointing to something with finger”. For each instance of the dataset, the 3D joint coordinates of the full body are detected by Kinect depth sensors for all the frames. The performance of a model for this dataset is evaluated for two benchmarks. The Cross-View (CV) benchmark measures the performance of the model for viewpoints the model was not trained with. Similarly, the Cross-Subject (CS) benchmark measures the performance of the model for subjects the model was not trained with. The accuracy rates of the models are presented collectively in Table 2.3 and the highest accuracy rate of each benchmark is highlighted. The top-performing model for both the CV and the CS benchmark is [25]. Moreover, the performance of [24] is impressive given the reduced computational complexity of the model.

2.3. Time-series classification

A common task when working with time-series data is classification. For this task, the features of a time-series are fed into a ML model which predicts the class of the input. Subsequently, we describe good practices for the time-series classification task based on [23].

“Traditional” ML models are not built around the idea of temporal notions. Instead, this concept of time is completely absent from the models. Therefore, in order to apply a ML algorithm to time-series data, one must find a way to make more general methods interpret temporal data. Selecting the data of the time-series as features and feeding these features to a ML model is not a solution, since different time-series might not have the same length and, most importantly, due to the dynamic nature of the data. For example, if the global maxima of the time-series of the training set usually occur at time-steps 4,5,7,8, then a traditional ML model might not be able to classify correctly a time-series whose global maximum occurs at time-step 6. Therefore, feature generation is necessary as a preprocessing step in order to encode the temporal evolution of the data. To this end, Lubba et al. [22] propose a set of canonical features that can be generated for every time-series, regardless of its nature, to achieve satisfactory performance. Simple temporal statistics, linear autocorrelation, nonlinear autocorrelation, the longest period of consecutive values above the mean, etc are listed as features of this set. Nevertheless, in most cases domain knowledge or a thorough investigation of the dataset can lead to better results than a generic feature set. The more one knows about the data, the more meaningful features he can generate.

Regarding DL models, while a Fully Connected DL model may perform better than a traditional ML model depending on the dataset, the Fully Connected model is also not time-aware. Therefore, similar to the traditional ML models, feature generation is necessary as a preprocessing step in this case. On the other hand, other DL architectures such as RNNs, CNNs and GCNs can capture the evolution of the time-series data on the temporal axis. In these cases, one can feed the time-series data directly to the DL models.

3. METHODOLOGY

3.1. Dataset

The dataset used for this thesis consists of the RGB and the depth information of human movements where a person reaches for an object, grasps it and moves it left of its initial position (based on [48]). This information was collected in [51] and was available in rosbags[2]. Each rosbag contained one movement and was named based on the object the person reached for, the anonymization code of the participant and the number of the repetition. A more detailed description of the dataset is provided in Section 3.1.1. Section 3.1.2 presents the methodology followed in this thesis to extract the skeletal data from the RGB-D information of the movements.

3.1.1. Movement Description

The participant sat on a height-adjustable chair so that the wrist and the elbow of his right upper limb formed an approximately right angle with his shoulder. In the beginning, the participant's right hand thumb and index touched each other over a predefined starting point marked on the table. One of three different objects was placed on the table, 63cm away from the participant's body in a perpendicular direction (Img. 3.1). The participant was asked to reach, grasp the object and move it 30cm to the left of its initial position, at a marked final position. The hand movement was performed in a natural way.

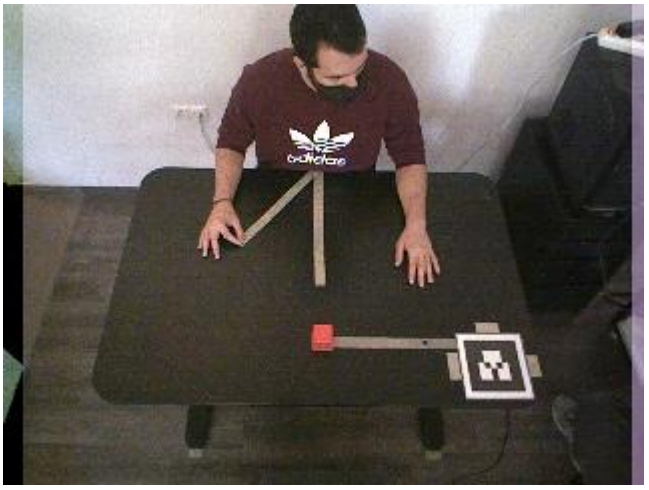

Image 3.1: The participant's initial posture in the experiment's workspace

The three objects used had the same cubical shape, but different sizes (Img. 3.2). Specifically, the participants reached for the following objects:

- a small-sized cube with a 2.5 cm edge length
- a medium-sized cube with a 5.5 cm edge length
- a large-sized cube with a 7.5 cm edge length

The movements were performed by eight right-handed participants Each participant reached for each one of the objects 30 times. Therefore, in total, 720 reach-to-grasp movements were contained in the dataset.

[2] http://wiki.ros.org/rosbag

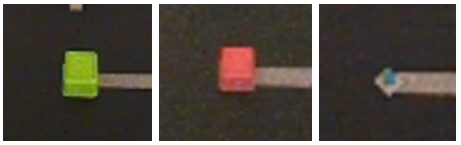

Image 3.2: The large, medium and small cubes.

Each movement was recorded using an ASUS Xtion Pro camera with a frame rate of 60 FPS. The camera was oriented almost vertically to the plane of the table throughout the data collection process (bird-eye-view).

3.1.2. Pose Estimation

To extract the kinematic features needed for object prediction (as in [48]) the 3D coordinates of the hand keypoints had to be estimated. After reviewing multiple state-of-the-art RGB-based and depth-based 3D hand pose estimation algorithms (Section 2.1), a ROS pipeline[3] based on OpenPose [50] was chosen to be used. Figures 3.1 and 3.2 show the skeletal data of the participant's body and hands accordingly that were extracted using the OpenPose framework.

In this ROS pipeline, after the extraction of the 2D skeletal data of a movement, the 2D coordinates were lifted to the 3D space by combining the 2D detections with the depth map information. After careful inspection of the 3D movement dataset, it was observed that depth information resulted in amplifying the noise of the dataset and, also, in information loss, because a significant percentage of the depth map values were undefined. *As a result, the dataset which was finally decided to be used for the classification analysis, contained the* **2D skeletal data** (*OpenPose pixel keypoints) of the movements*.

More specifically, the dataset included the following information:

- the sequence number of each frame
- the timestamp information of each frame
- a probability as the confidence value of each of the keypoints' detections for every frame
- the x- and y-coordinates of each of the keypoints' detections for every frame (measured in pixels)

While the frame rate of the ASUS Xtion Pro camera was 60 FPS, the inference rate of OpenPose was around 21.3 Hz for a Quadro RTX 4000 GPU, which was our best available hardware in term FLOPS. Therefore, OpenPose was not used in an online, real-time manner in order to extract the skeletal data for every frame and to avoid information loss, given the duration of the movement of interest. Instead, the frames were processed offline using an appropriate ROS tool[4].

[3] https://github.com/Roboskel-Manipulation/openpose_3D_localization
[4] https://github.com/Roboskel-Manipulation/bag_read_service

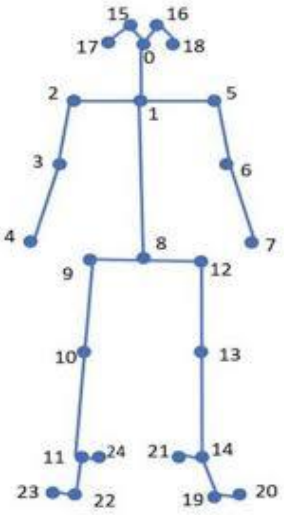

Figure 3.1: The full body keypoints of the OpenPose framework.

Figure 3.2: The hand keypoints of the OpenPose framework.

In the context of this work, only the right wrist of the full body (keypoint 4 in Fig. 3.1) and the thumb, index and middle tip of the right hand (keypoints 4, 8, 12 in Fig. 3.2) were taken into consideration. The other keypoints of the body and the hands were ignored, since they were not relevant to this work.

3.2. Preprocessing

While OpenPose is one of the predominant solutions for 2D body and hand pose estimation, this task is inherently difficult and, as a result, some of the predictions of the keypoints' locations were noisy. Removing this noise from the dataset could be integral for boosting the inference ability of a ML model.

Moreover, the full recorded movements, and, consequently, their extracted skeletal data, include a starting phase, where the participant's right hand is lying still on the table, a grasping phase, where the participant reaches towards the object with his right hand and grasps it and, finally, a moving phase, where the participant lifts the object and moves it. In order to predict which object will be grasped, the only information pertinent is that contained in the frames of the grasping phase.

Finally, to study the capability of a ML method to predict the size of the object in various phases of the grasping movement, from movement onset to object grasping, each movement had to be observed in various completion phases. As a result, data preprocessing is essential to extract the dataset's skeletal data which will be used later to engineer kinematic features. The preprocessing pipeline can be summarized into three steps (Fig. 3.3):

- **Filtering**: The frames where the keypoint detections of OpenPose were noisy were discarded.
- **Grasping movement phase identification**: The grasping phase of the movement was identified and the rest of the movement was discarded.

- **Grasping movement completion intervals**: The grasping movement was divided into five subsets (movement completion intervals) all starting from movement onset until the 20%, 40%, 60%, 80% or 100% of the movement completion.

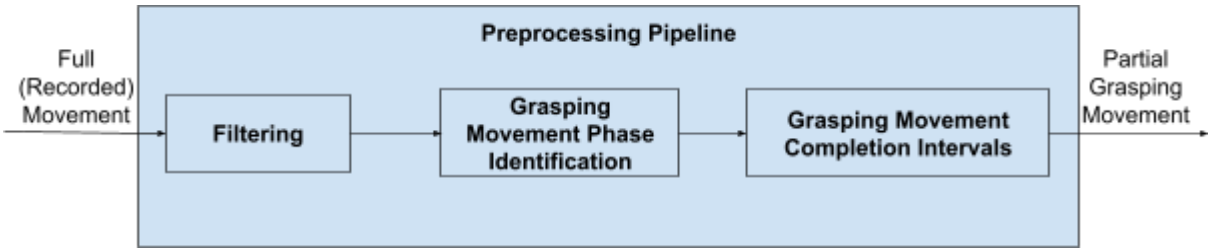

Figure 3.3: The preprocessing pipeline.

3.2.1. Filtering

In order to avoid erroneous results in identifying the grasping phase of the participants' movements and, as discussed above, to boost the ML models' inference ability by limiting the amount of noise in the kinematic features, frames with noisy keypoint detections were identified and filtered out. The filtering preprocessing was based on the **right wrist**, since this keypoint was later used for the grasping phase identification of the movement. Moreover, an extra advantage of wrist-based filtering was that the detections of the wrist had higher confidence values than the detections of other keypoints (Fig. 3.4) and, therefore, filtering out frames with noisy wrist detections would lead to less information loss than filtering out frames with noisy fingertips detections.

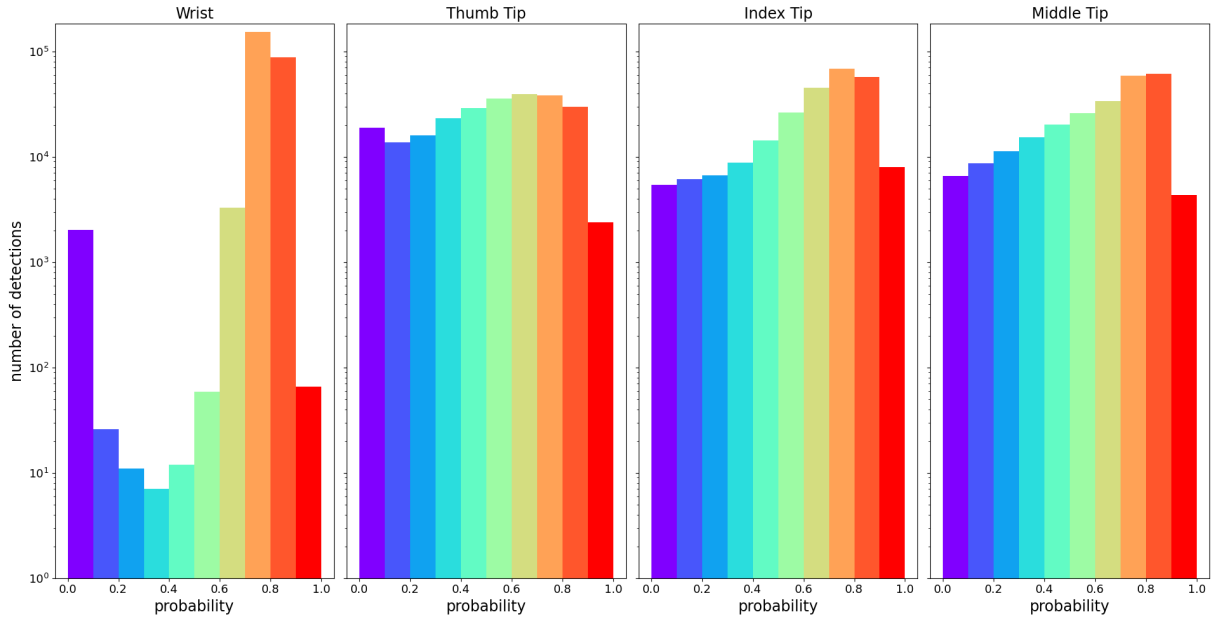

Figure 3.4: The histograms of the detection probabilities per keypoint.

After exploring the dataset, the criteria based on which a frame was filtered out were the following:

- the right wrist's probability was less than or equal to 0.6, or
- the minimum y-axis distance of a right wrist keypoint to the same joint keypoint of the previous and the next frame was more than 10 pixels.

Specifically, we both visually inspected the dataset (individual movements) and used the frequencies of the keypoint probabilities and the y-axis distances of the right wrist (Fig. 3.5). It was observed that the majority of the wrist detections had probability values larger than 0.6 and that actually a probability less than that was related to erroneous OpenPose output. Moreover, it is worth mentioning that the bin [0.0, 0.1] also had a high frequency. This is because when OpenPose completely failed to predict the position of a keypoint the probability, the x-coordinate and the y-coordinate of that keypoint were set to 0. Regarding the y-axis distances, a similar methodology was followed to define the corresponding criterion. In particular, the majority of the y-axis distances had small values and were concentrated in the [0,5] and [5,10] bins. In order to avoid filtering out frames whose previous frame had an outlier wrist y-coordinate, both the previous and the next frame were considered. In a real-time application, this criterion would cause a delay of one frame, which could probably be acceptable. Finally, a filtering criterion concerning the x-axis distances of the wrist was not implemented, because the grasping phase identification was based only on the y-coordinates of the wrist keypoint.

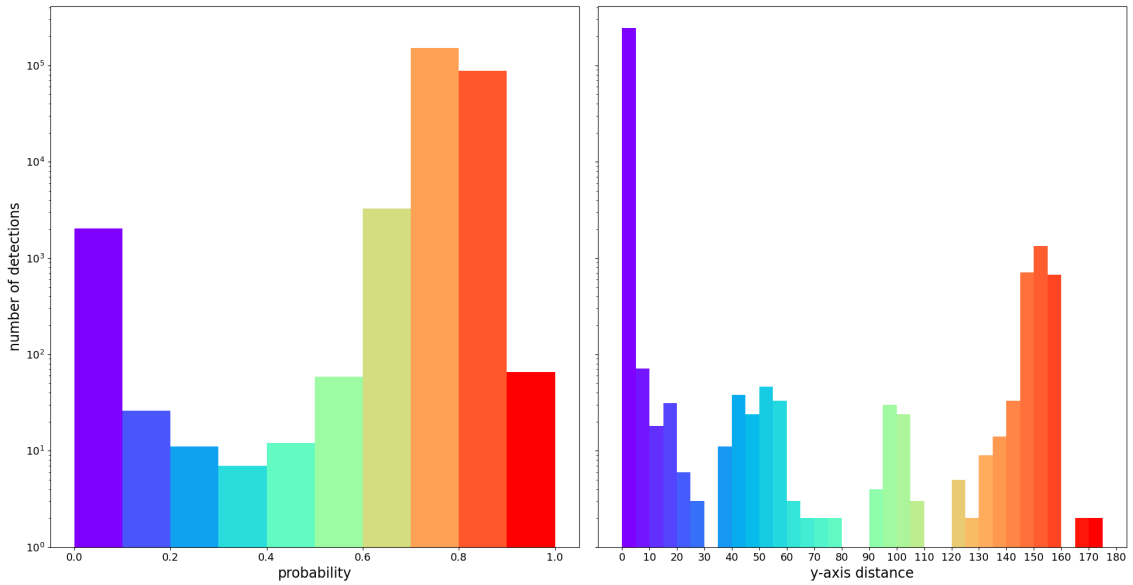

Figure 3.5: The right wrist histograms of the detection probabilities (left) and the y-axis distances between a keypoint and its neighbouring one (right).

An example of the filtering step of the preprocessing pipeline is presented in Figure 3.6a. In this subfigure, the wrist y-coordinates were plotted with respect to the frame sequence number for a full (recorded) movement of the dataset. The red points were identified as outliers based on the filtering criteria, while the blue points were identified as valid. The frames corresponding to the red points were discarded.

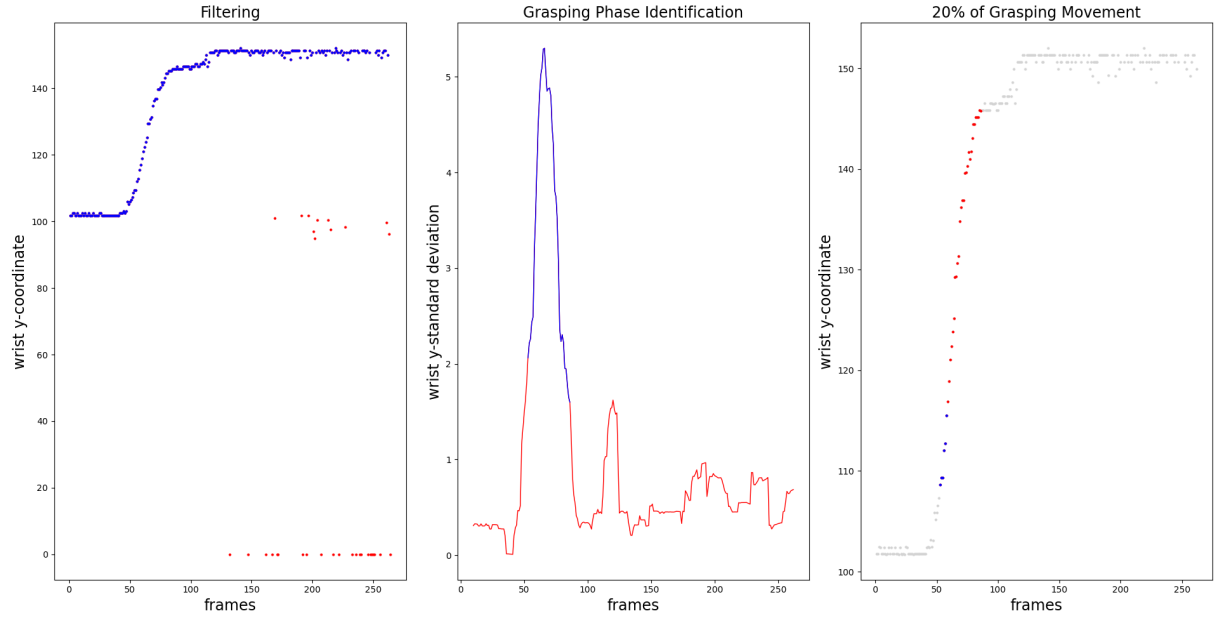

Figure 3.6: The preprocessing pipeline of a movement. a) Filtered movement (blue points) after removing the outliers (red points). b) Movement's grasping phase (blue trace) identification based on the standard deviations of the y-coordinates. c) 20% (blue points) of the movement's grasping phase (blue and red points).

3.2.2. Grasping Phase Identification

After filtering the noisy frames out, the frames related to the movement from its onset till the object grasping had to be identified and isolated from the rest of the data. The speed of the right wrist could potentially be used for this reason, as in [48]. However, calculating the instantaneous speed of a keypoint was not particularly helpful in our case, because of the noise of the data. Instead, the standard deviation of the right wrist y-coordinates over a window of 10 frames was used as an approximate measure of the speed. The dispersion of the wrist positions was measured only in the y-axis, since in this experimental setup the participant moves his hand mainly in the y-axis direction during the grasping phase. The window size was fixed and the 10 frames of the window did not necessarily have consecutive sequence numbers due to the filtering preprocessing step.

The criteria based on which the grasping phase of the movement was identified were the following:

- **Movement onset**: The moment when the standard deviation becomes greater than 2.0 pixels for the first time. The last frame of the window was considered as the first frame of the movement's grasping phase and, consequently, the movement before this frame was discarded.
- **Movement end**: The moment when the standard deviation becomes less than 1.5 pixels for the first time after the standard deviation's global maximum. The penultimate frame of the window was considered as the last frame of the movement's grasping phase, while the last frame of the window was considered as the first frame after the grasping phase of the movement had ended. Consequently, the movement information of the last frame of the window and of the subsequent frames were discarded.

These criteria were chosen experimentally, by visually inspecting plots like Fig. 3.6b-c for all the movements. Regarding the wrist y-standard deviation plot (Fig. 3.6b), the goal was to include in the grasping phase only the frames in the bell-shaped segment of the curve. Likewise, for the wrist y-coordinate plot (Fig. 3.6b), the goal was to include only the frames of the segment with the steep slope. Complementary to the aforementioned goals, the limit of the movement onset criterion was set high enough in order to avoid starting a movement prematurely due to noise. Similarly, the limit of the movement end criterion was set low enough to avoid stopping a movement prematurely due to noise.

The movement onset criterion can be implemented for a real-time application. On the other hand, the movement end criterion is not applicable in real-time, since the global maximum of the y-coordinates' standard deviation is not known before the end of the full recorded movement. Moreover, the chosen criteria are dependent to a large degree on the configuration of the dataset's workspace and would have to be revised for an application where initial positions of the hand and target object differ. For more configuration-independent grasping phase identification criteria, the standard distance of the (x,y)-coordinates could be used instead of the standard deviation of the y-coordinates.

An example of the grasping phase identification step of the preprocessing pipeline is presented in Figure 3.6b. In this subfigure, the standard deviation of the right wrist y-coordinates was plotted with respect to the frame sequence number of the last frame of the window. Based on the movement onset and end criteria, the frames of the blue trace were identified as the frames of the grasping movement.

3.2.3. Grasping movement completion intervals

In a HRC setting, the robot should be able to predict which object the human would grasp before his movement is completed. To evaluate the performance of the prediction methods at different completion intervals of the grasping movement, the movement was divided into five completion parts. Specifically, the grasping phase of the movements was divided into five subsets that included the 20%, 40%, 60%, 80% or 100% of the movements. For each of these cases, the available information was associated only with the frames whose movement completion percentage was less than or equal to 20%, 40%, 60%, 80% or 100% respectively. A movement completion percentage was calculated for each of the frames of the grasping movement. The calculation was based on the frames' sequence number rather than the their timestamps according to the following formula:

$$\frac{seq\_no(x) - seq\_no(1) + 1}{seq\_no(L) - seq\_no(1) + 1},$$

where $seq\_no(i)$ is the sequence number of the i-th frame, the grasping phase of the movement has L frames in total and the movement completion of the x frame $(1 \leq x \leq L)$ is calculated. Figure 3.7 shows the number of frames contained in each completion interval across the whole dataset.

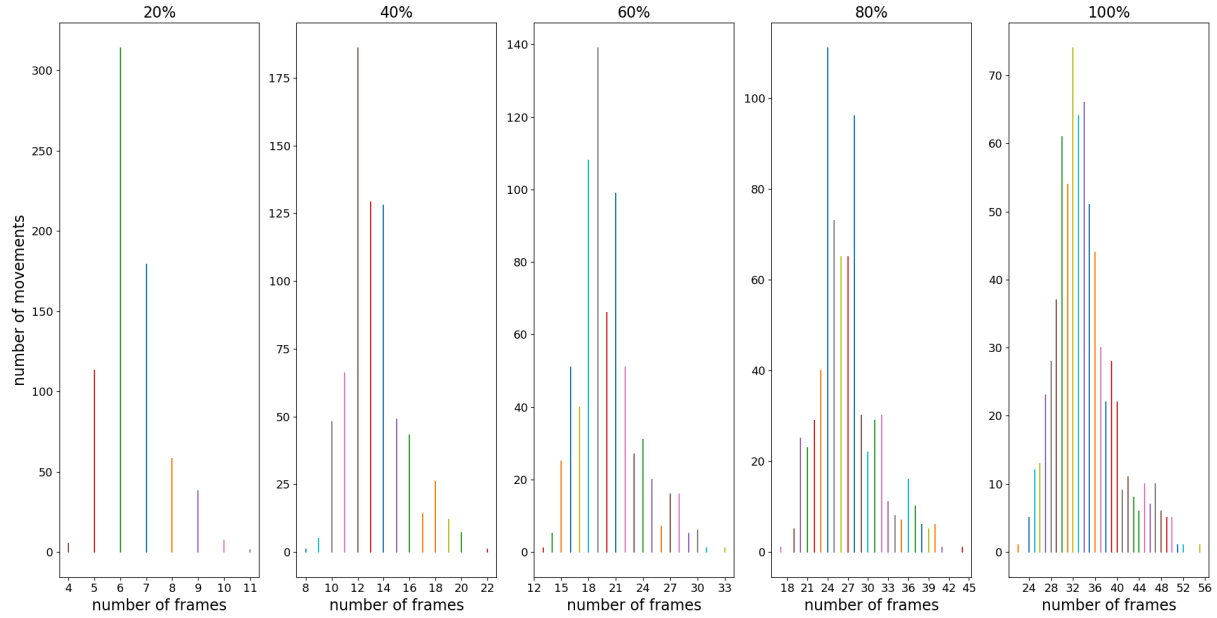

Figure 3.7: The number of frames contained in the 20%, 40%, 60%, 80% and 100% intervals of the grasping movement (left to right).

An example of the grasping movement completion intervals step of the preprocessing pipeline is presented in Figure 3.6c. In this subfigure, the wrist y-coordinates were plotted with respect to the frame sequence number, similar to 3.6a. The blue points correspond to the 20% movement completion interval of the grasping movement, while the combination of blue and red points corresponds to the full grasping movement. The gray points correspond to frames, which were identified as valid during the filtering step, but were not included in the grasping movement. Finally, the outlier points of the filtering step were not plotted.

3.3. Feature Engineering

After the preprocessing pipeline was applied to all of the dataset's instances, five movements where the data collection was problematic were identified and excluded from the dataset. The problematic data collection was verified by checking the respective rosbags of these movements. The problems that were encountered in the rosbags were the following:

- **Problematic detection of movement onset**: There was only a very brief pause or no pause at all at the beginning of the recording (3 movements).
- **Missing information**: The grasping phase of the movement was not recorded at all (2 movements).

For the 715 valid movements resulting from the preprocessing pipeline, the following features were examined in this work:

- **thumb-index aperture** (thumb tip - index tip distance)
- **thumb-middle aperture** (thumb tip - middle tip distance)
- **index-middle aperture** (index tip - middle tip distance)
- **wrist x-coordinate**
- **wrist y-coordinate**
- **wrist xy-standard distance**

- **wrist x-standard deviation**
- **wrist y-standard deviation**
- **wrist xy-speed**
- **wrist x-speed**
- **wrist y-speed**

3.3.1. Feature Filtering

During a movement's preprocessing and, more specifically, during the filtering step, noisy frames were identified and filtered out. This preprocessing filtering step was based on the probabilities and the y-axis distances of the wrist keypoint detections. In Fig. 3.8-3.10 the probability, the x-axis distance and the y-axis distance histograms of the fingertip keypoints were plotted before and after the first two stages of preprocessing (for the 100% movement completion interval). Following a comparison of the distributions of the fingertips' histograms before and after the preprocessing in these figures, it was observed that a large percentage of the fingertips' detections with loperw confidence values or large distance values were discarded. However, some of the remaining detections of the thumb, index and middle tip contained noise even after the preprocessing pipeline was applied. This can be observed in the "after" probability histograms of Fig. 3.8-3.10, since multiple predictions have low confidence values and, also, in the "after" distances histograms of the same figures, since there are multiple outliers with large distance values (higher than 10 pixels) either in the x- or the y-axis. These noisy detections would then result in noisy aperture feature values, and were thus identified and filtered out during the feature filtering step.

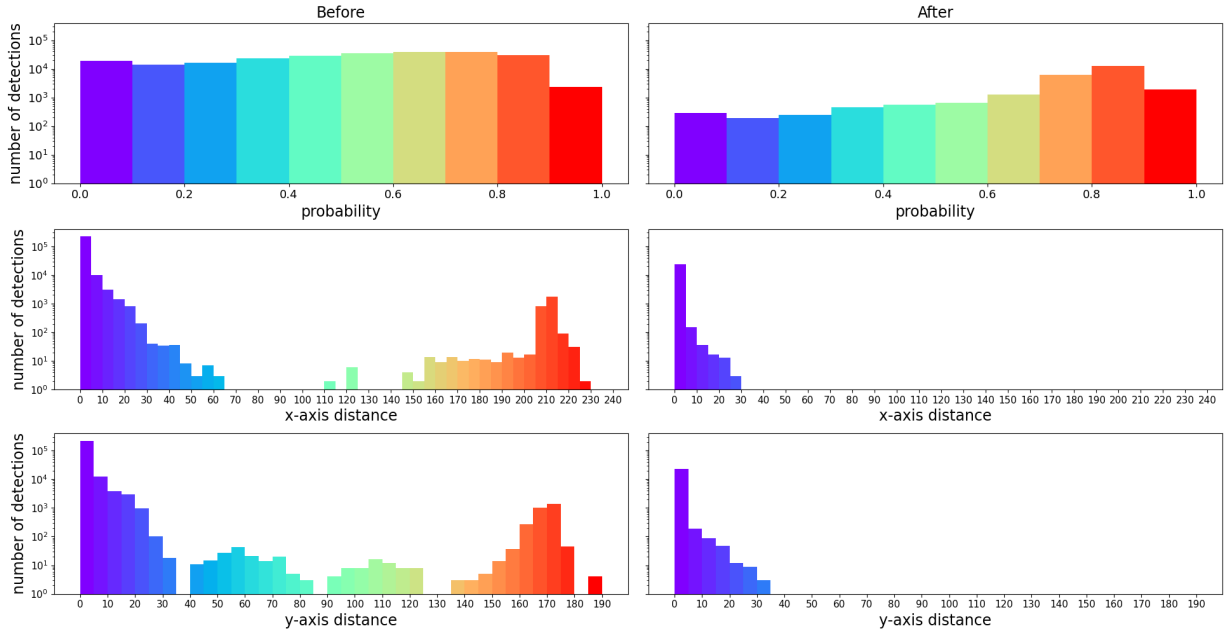

Figure 3.8: The probability, x-axis distance and y-axis distance histograms for the thumb fingertip keypoint before (left) and after (right) the preprocessing of the wrist key points.

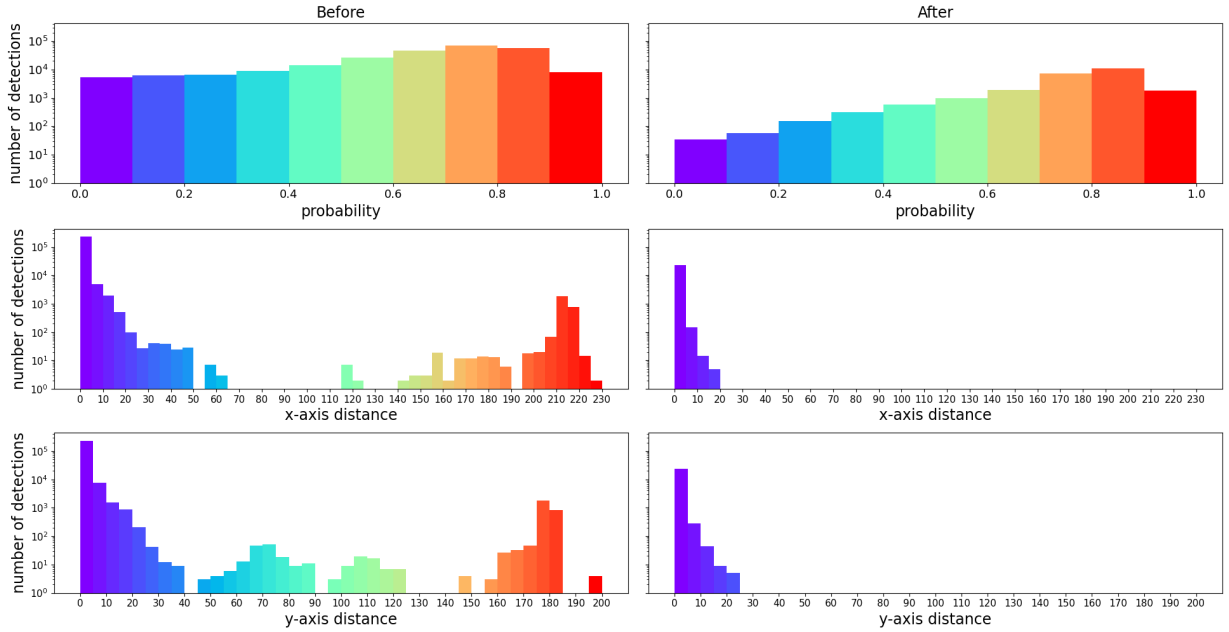

Figure 3.9: The probability, x-axis distance and y-axis distance histograms for the index fingertip keypoint before and after the preprocessing of the wrist key points.

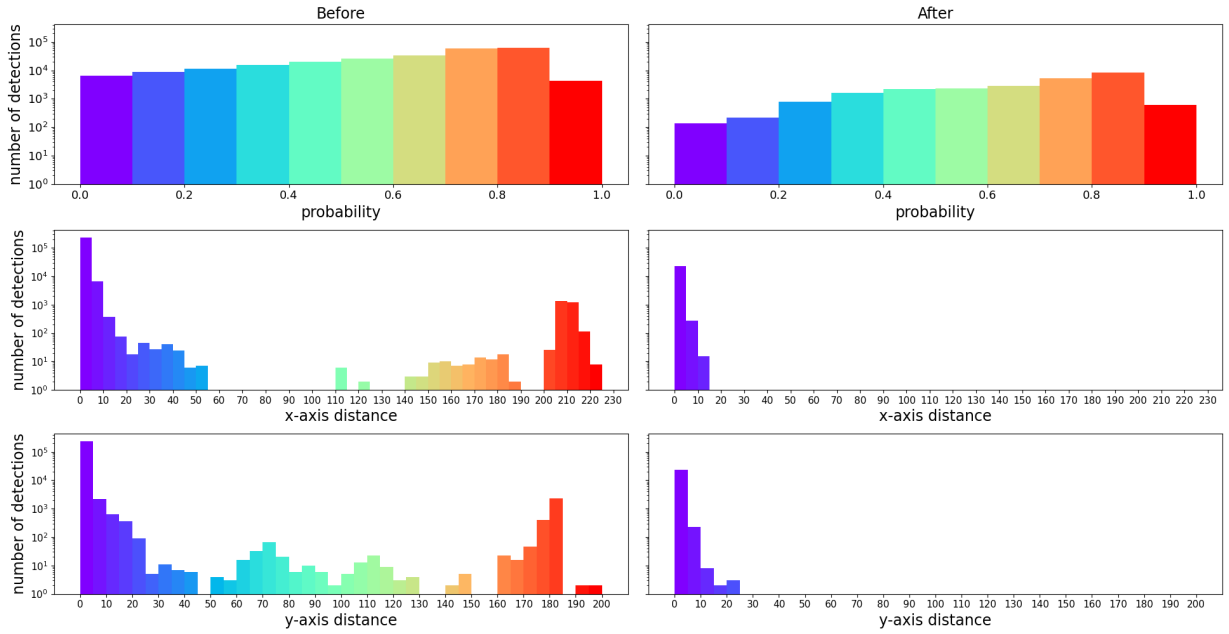

Figure 3.10: The probability, x-axis distance and y-axis distance histograms for the middle fingertip keypoint before (left) and after (right) the preprocessing of the wrist key points.

In order to identify whether an aperture feature value was noisy, we first defined that a keypoint detection was noisy when:

- the probability of the detection was less than or equal to 0.6,
- the minimum x-axis distance from the keypoint to the same joint keypoint of the previous and the next frame was more than 10 pixels, or
- the minimum y-axis distance from the keypoint to the same joint keypoint of the

previous and the next frame was more than 10 pixels.

These criteria were selected similarly to the corresponding preprocessing noise detection criteria for the wrist keypoint. In contrast to the latter criteria, the x-axis distances were also considered during feature filtering, since the aperture features depend on both the x- and y-coordinates of the fingertips, while the grasping phase identification of the preprocessing pipeline was based exclusively on the y-coordinates of the wrist. Fig. 3.8-3.10 show that, after the preprocessing pipeline was applied, the distance bins with the majority of the predictions contained distances with values less than 10 pixels. Moreover, the probabilities were distributed in a near uniform manner and, therefore, values lower than a certain threshold could not be easily excluded as outliers. As a result, a confidence value of 0.6 was selected as the threshold of the probability criterion based on the corresponding preprocessing wrist evidence. This evidence indicated that probability less than 0.6 was related to erroneous keypoint detections.

Based on the aforementioned criteria, a feature which was calculated based on two keypoints was considered to be noisy, when either one of the keypoints was noisy. More specifically:

- a thumb-index aperture feature was identified as noisy, when the thumb fingertip or the index fingertip detection was noisy.
- a thumb-middle aperture feature was identified as noisy when the thumb fingertip or the middle fingertip detection was noisy.
- an index-middle aperture feature was identified as noisy, when the index fingertip or the middle fingertip detection was noisy.

After filtering these values out, the percentage of the valid thumb-index aperture features was 83.6%, the percentage of the valid thumb-middle aperture features was 62.6% and, finally, the percentage of the valid index-middle aperture features was 66.3%.

One notable difference between the filtering methodology during the preprocessing and the feature engineering phase was that, in the former stage, if a wrist keypoint value was identified as an outlier all the information regarding the frame was discarded, while, in the latter stage, if a kinematic feature value was identified as an outlier only this particular value was discarded.

The wrist feature values were not filtered, since:

- the frames with noisy wrist y-coordinates or detection probabilities were already filtered out during preprocessing.
- after visually inspecting Fig 3.11, one can observe that the wrist's x-coordinates did not contain noise after the preprocessing pipeline was applied.

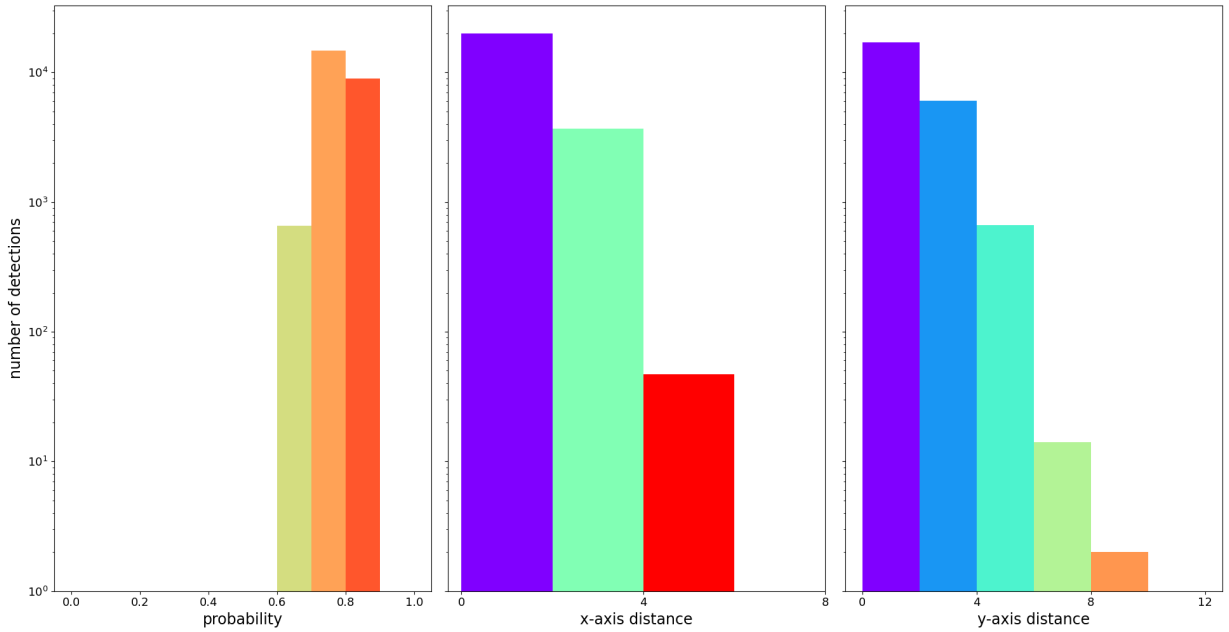

Figure 3.11: The probability, x-axis distance and y-axis distance histograms after the preprocessing of the wrist keypoints.

3.3.2. Features: Definitions, Motivations and Discernibility Analysis

The kinematic features, which were examined in this thesis, were selected either based on existing literature [48] or by observing the RGB-D videos of the movements. Furthermore, in order to assess qualitatively each feature, Fig. 3.12-3.15 were used.

These figures were used in order to evaluate the discernibility of each kinematic feature; the figures were visually inspected in order to assess whether there were observable differences between the classes for each feature. To plot these figures, the features of each movement were first evaluated with respect to the frames' movement completion percentages. Following this, the necessary values for the plots of a kinematic feature were calculated for 0%, 20%, 40%, 60%, 80% and 100% of the movements in the following way:

A. If a frame occurred on exactly the 0%, 20%, 40%, 60%, 80% or 100% of a movement completion time and the feature value of that frame was not filtered out, this feature value was used for the corresponding movement completion percentage.

B. If none of the movement's frames occured on one of the aforementioned movement completion percentages or if the feature value was filtered out, the previous and the next frame with unfiltered feature values were used. The feature values of these two frames were linearly interpolated for that movement completion percentage.

C. If a previous or a next frame with valid feature values did not exist, the feature value of the closest frame was used instead of linear interpolation.

D. If all the frames of the movement had their values filtered out, the feature values were not calculated for any of the percentages.

Finally, the mean μ and the standard deviation σ were calculated based on all the

movements (of all participants) of the same label for each of these percentages. In the case of the aperture features, whose values were filtered during the feature engineering phase, movements without any valid feature values were not taken into consideration for the corresponding feature plots (D). Consequently, out of the total 715 movements of the dataset, 713 movements were considered for the thumb-index aperture plot, 655 movements were considered for the thumb-middle aperture plot and, finally, 644 movements were considered for the index-middle aperture plot. On the contrary, all of the 715 movements were taken into consideration for the wrist features plots, since filtering was not applied to these features during feature engineering. The plotted curves of the figures follow the mean μ of the movements, while the shaded areas contain all the points in [μ-σ, μ+σ].

3.3.2.1. Fingertips Aperture Features

The fingertips aperture features were defined based on the Euclidean distance metric. More specifically, the thumb-index aperture was defined as the Euclidean distance between the thumb and the index fingertip detections. Likewise, the thumb-middle aperture was defined as the Euclidean distance between the thumb and the middle fingertip detections and the index-middle aperture as the Euclidean distance between the index and the middle fingertip.

While the participants moved towards the object, their hand configuration adjusted to the target object properties. The aperture features were used to describe the hand configuration and, more specifically, the relative positions of the fingertips. In particular:

- The *grip aperture or thumb-index aperture* is a kinematic feature that has been shown to provide information for the prediction of size of the object to be grasped [48]. Similarly to [48], Fig. 3.12a shows that the participants gradually open their grip since the start of the movement, before, finally, conforming their grip to the size of an object. The comparison between the movements of the different classes (Fig. 3.12a, comparison column) shows that one could possibly discern the class of the object based on this feature.

After inspecting the videos of the movements, it was noticed that some participants used only their thumb and index fingers to grasp the small object, while the thumb, index and middle fingers were used to grasp the medium and the large object. Therefore, the thumb-middle and index-middle apertures were also taken into consideration.

- After inspecting Fig. 3.12b, one can notice that the *thumb-middle finger aperture* evolves over time similar to the thumb-index aperture. More specifically, the larger the target object, the greater the distance between the thumb and the middle fingertips is. Also, similar to the thumb-index aperture, the feature can be useful from the beginning of the movement and its discernibility improves over time. This similar behaviour can be explained, since the distance between the index and the middle fingertips is significantly less than the distance between the thumb and the index independently of whether the middle finger is used to grasp the object or not.

- Regarding the *index-middle aperture feature*, one can observe that the standard deviation increased after 60% of the movement for the medium and the large object (Fig. 3.12c). This can be explained, as an inconsistent behavior was

exhibited across different participants; not all of them used their middle finger in the same way when grasping the medium and the large object. Therefore, if a ML model was trained with other movements of the participant, the variability of the feature could prove useful either on its own or in conjunction with a more consistent feature across participants. Furthermore, based on the comparison column of the figure, the index-middle feature could be potentially used at the later stages of the movements (80% 100%) to discern the large object from the medium one. In any case, the discernibility of the other aperture features seems to be superior.

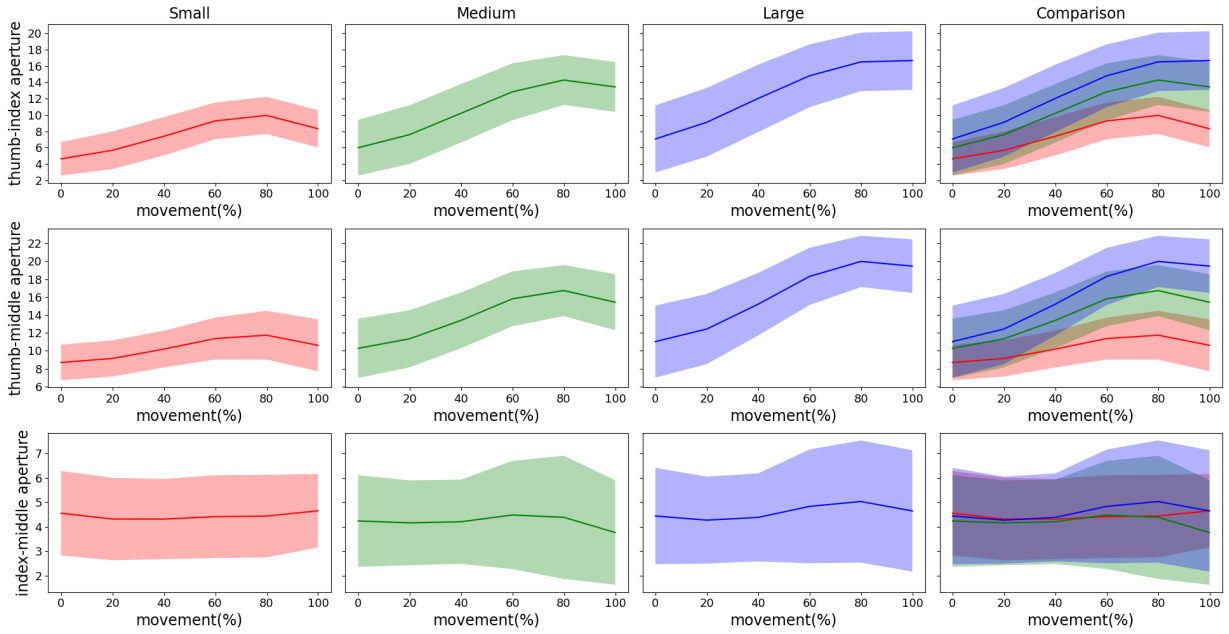

Figure 3.12: The (a) thumb-index, (b) thumb-middle and (c) index-middle aperture features with respect to the movement completion percentage (%) and the target object.

3.3.2.2. Wrist Coordinate Features

Regarding the wrist x- and y- coordinates features, the OpenPose wrist detections were used without any further processing.

These features were selected to capture the evolution of the movement in space. The 2D coordinates were used instead of the wrist's 3D coordinates, which were utilized in [48]. Based on Fig. 3.13, there are not any significant observable differences both for the wrist x- and y-coordinates among the small-, medium- and large- labeled movements. It can be argued, however, that the wrist coordinates in conjunction with time could be useful in order to predict the target object. This is due to the fact that the smaller an object is, the slower one might move to achieve the more accurate grip needed for a small object (Fitts Law) [52]. Therefore, the participant's hand movement would follow the same coordinates to grasp different objects, but at a different time. As a side note, a downside of the wrist coordinate features is that, if used, a ML model would not be able to generalize to different workspace configurations without first having been trained with additional movement data, where the hand and the target object are placed in multiple different initial positions.

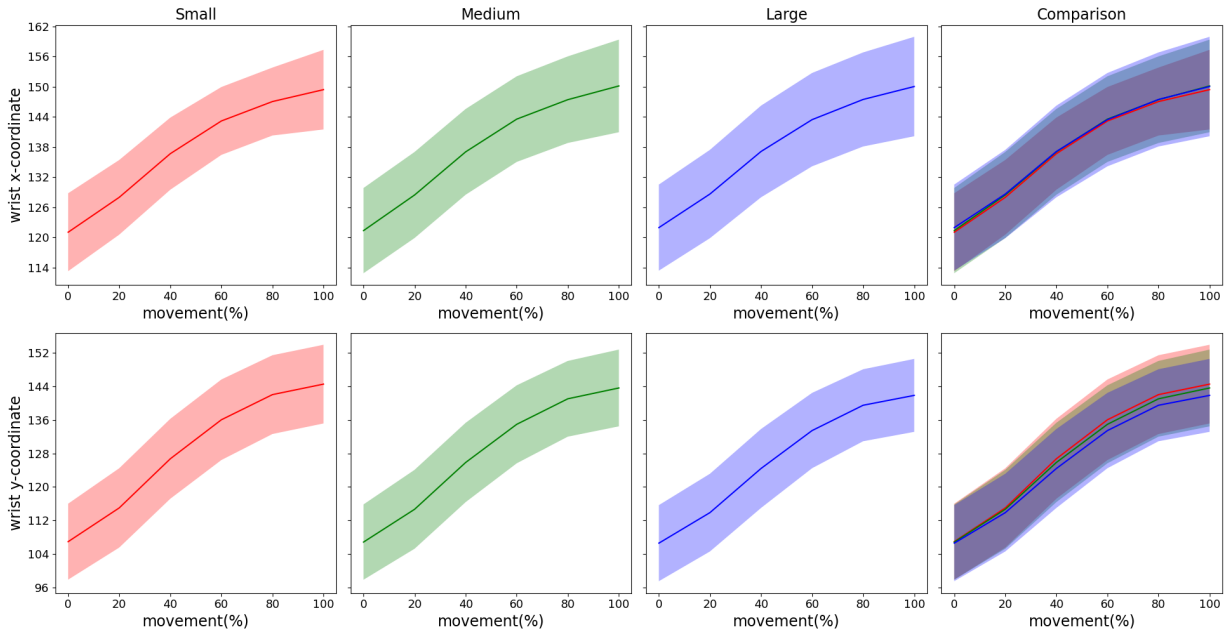

Figure 3.13: The wrist x- and y-coordinate features with respect to the movement completion percentage (%) and the target object.

3.3.2.3. Wrist Instantaneous Speed Features

The wrist's instantaneous speed for a frame was calculated based on the current and the previous frame. In general, the instantaneous speed is defined as the division of the instantaneous spatial shift by the instantaneous temporal shift. The instantaneous temporal shift was calculated as the difference between the timestamps of the frames, while the instantaneous spatial shift was calculated based on the Euclidean distance between the wrist keypoint detections of the two frames. More specifically, the instantaneous spatial shift for the wrist x-speed feature was calculated as the Euclidean distance between the x-coordinates of the wrist keypoint detections. Likewise, the instantaneous spatial shift for the wrist y-speed and the wrist xy-speed were calculated as the Euclidean distance between the y-coordinates and the two-dimensional (x,y)-coordinates of the wrist keypoint detections respectively.

In [48], the wrist speed was used as a kinematic feature with the expectation that the larger an object is, the later the speed's global maximum occurs. In the case of [48] the speed was calculated in m/s based on the 3D wrist coordinates of the wrist's marker, while in this thesis, the wrist speed was measured in pixels/s based on the 2D wrist detections. Nevertheless, Fig. 3.14 suggests that in our case, one cannot discern the target object based on wrist speed. A plausible explanation for this could be either the features' noise or the fact that the objects' size differences are too small to expose the phenomenon. These two explanations are not mutually exclusive. The noise of the speed features is evident in the figure, since the plotted curves are not bell-shaped as they would be in the absence of noise. The large standard deviations of the speed features are also supporting evidence towards this direction. Nevertheless, keep in mind that inconsistency between the participants could also be the reason for the large standard deviations. As a result, in order for a ML model to predict the target object of a movement based on the speed features, the model would have to learn patterns that are not apparent in the representations of Fig. 3.14, if any. Finally, it is worth noting that, while using a combination of the wrist-x and wrist-y speed features may be more informative than the wrist-xy speed feature, the latter is more independent of the

workspace configuration. Therefore, a ML model could generalize more easily to different workspace configurations when only the latter feature is considered instead of the two former ones.

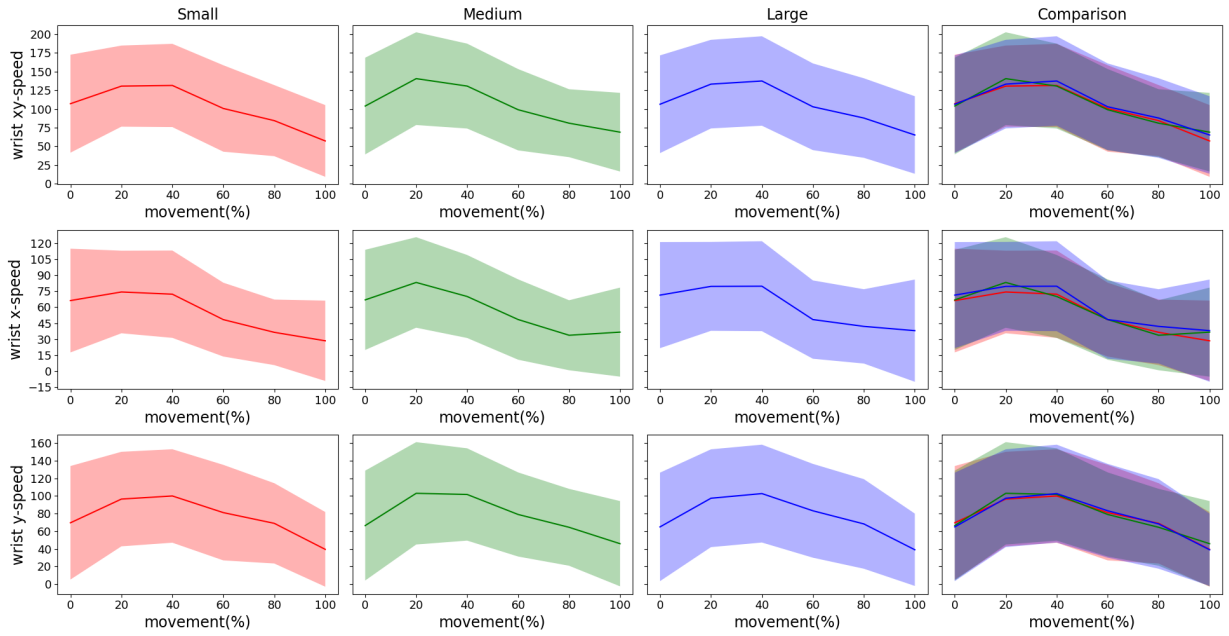

Figure 3.14: The wrist xy-, x-, and y-speed features with respect to the movement completion percentage (%) and the target object.

3.3.2.4. Wrist Coordinates' Dispersion Features

During the grasping phase identification step of the preprocessing pipeline, the wrist coordinates' dispersion in the y-axis was used as an approximation of the wrist's speed, since, if the wrist detections did not contain any noise, the instantaneous speed's curve would be similar to the dispersion's bell-shaped curves. Similarly, during feature engineering, the wrist coordinates' dispersions were measured in the xy-plane, the x-axis and the y-axis as an alternate way to measure the wrist's speed (instead of the wrist xy-speed, wrist x-speed and wrist y-speed features respectively).

The wrist xy-standard distance, x-standard deviation and y-standard deviation features measured the dispersion of the wrist coordinates in the xy-plane, the x-axis and the y-axis respectively. The dispersion of the wrist coordinates was calculated within a window of 10 frames and this value was assigned as an approximation of the hand's speed to the last frame of the window (similar to the grasping phase identification step of the preprocessing phase). When an approximation of the x-axis or the y-axis wrist speed was calculated, the standard deviation definition was applied to measure the dispersion of the coordinates in the corresponding axis. When an approximation of the wrist speed in the xy-plane was calculated, the standard distance definition was applied to measure the dispersion around the points' centroid in the xy-plane. The standard distance metric was calculated with the following formula:

$$SD = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{X})}{n} + \frac{\sum_{i=1}^{n}(y_i - \bar{Y})}{n}}$$

The dispersion of the 2D wrist coordinates can be represented in a two-dimensional

plane by drawing a circle with the centroid of the points as its center and with its radius equal to the standard distance value.

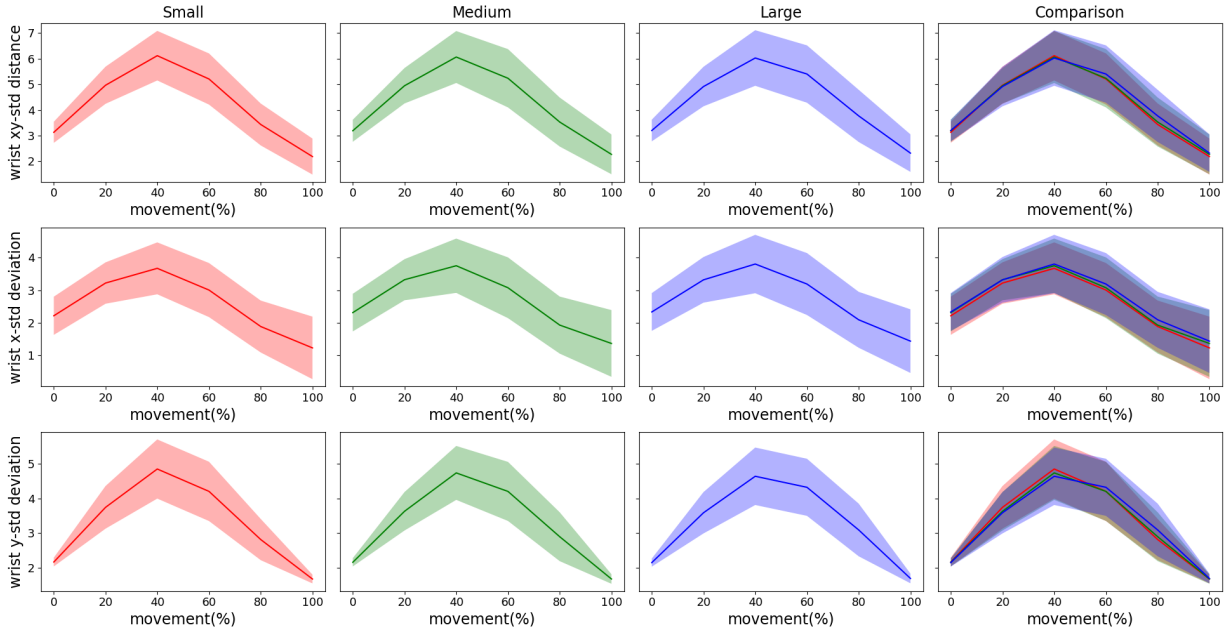

Figure 3.15: The wrist xy-standard distance, x-standard deviation and y-standard deviation features with respect to the movement completion percentage (%) and the target object.

Based on Fig. 3.15 one can observe that the dispersion features follow a bell-shaped pattern. Nevertheless, in the figures of both the wrist coordinates' dispersion and the wrist instantaneous speed features, one cannot discern the target object based on the movement completion percentages of the curves' global maxima. In both of these cases, the global maxima occur at approximately the same normalized time point, independently of the target object. As mentioned for the wrist instantaneous speed features, the reason for this phenomenon could be either noise or the minor size differences between the objects. Furthermore, similar to the wrist's speed features, a ML model could generalize more easily to different workspace configurations when only the wrist xy-standard distance feature is considered instead of a wrist x- and wrist y-standard deviation features combination.

3.4. "Traditional" Machine Learning approach

After preprocessing and feature engineering, two partial movements did not have the same number of frames in the general case and, as a result, they were not represented in the same way. In order to train a ML model, a common representation between the dataset's movements was necessary. Moreover, even if each movement had the same number of frames T, a one-dimensional $T \cdot F$ movement representation in which the F engineered features of all the frames were concatenated, would not be helpful for a ML model to perform satisfactorily. The reason why this model would not yield competent results is that it would not have captured any temporal dependencies; swapping two features which correspond to different frames in the movement representation would not affect the performance of the model.

To remedy both of these problems, each (partial) movement was considered as a

multivariate time-series and the following meta-features were extracted for each of the engineered kinematic features of the movements:

- **Mean**: the mean of the feature values of the partial movement.
- **Standard deviation**: the standard deviation of the feature values of the partial movement.
- **Max**: the global maximum of the feature values.
- **Min**: the global minimum of the feature values.
- **tmax**: the absolute time when the global maximum of the feature occurs.
- **tmin**: the absolute time when the global minimum of the feature occurs.
- **Slope**: the slope of a line which is calculated with linear least-squares regression based on the absolute time and the feature values of the partial movement.

Moreover, the **number of frames** was added as a feature of each movement.

For the tmax, tmin and slope meta-features the absolute time of the movement was used. More specifically, for each of the movements, the time of the first frame was set to 0, while the time of the following frames was calculated based on the timestamps as the difference between the timestamps of the first and the current frame of the movement. This approach was followed instead of using the movement completion percentage as time, since in a real-time application the movement completion percentage of each frame would not be known until the end of the movement.

In the case of the aperture features, due to feature filtering, a partial movement might not have any valid feature values. In this scenario, all the meta-features of this particular feature could not be defined and were set to 0. Similarly, in case the partial movement contained only one valid feature value, the slope meta-feature could not be defined and was set to 0.

After the meta-feature extraction, each movement could be represented in the same way, as a set of $F \times M$ features - meta-features combinations, where F is the number of features and M is the number of the meta-features. These representations captured each feature's summary statistics (mean, standard deviation), peaks (max, min), temporal peak information (tmax, tmin) and spatiotemporal information (slope). Finally, the number of frames also provided temporal information regarding the partial movement.

Following this "Traditional" Machine Learning approach, the following Machine Learning models were evaluated:

- **Random Forest (RF)**
- **Gradient Boosting (GB)**
- **Extra Trees (ET)**
- **Support Vector Machine (SVM)**
- **Gaussian Process (GP)**

To this end, the Scikit-learn [41] Machine Learning framework was used.

3.5. Deep Learning approach

For the Deep Learning approach, the same problem which was encountered in the "Traditional" Machine Learning regarding the different number of frames per (part of the) movement, had to be solved. In contrast to the previous approach, however, a convolution operation could be used for a Deep Learning model to capture temporal dependencies between the feature values of consecutive frames. Therefore, a simple solution of zero-padding was used to represent each partial movement in the same way. For this purpose the maximum number of frames was first calculated for the 20%, 40%, 60%, 80% or 100% of the movement's completion. Afterwards, each movement was represented as 1D pseudo-image with the following properties:

- The pseudo-image had as many 1D channels as the kinematic features which were used.
- Each channel had n_t values; n_t is defined as the maximum number of frames of all the partial movements with the same movement completion percentage.
- If a partial movement consisted of f frames, the f values of each kinematic feature were sorted in an ascending chronological order in its corresponding channel.
- The remaining $n_t - f$ values on the right of each channel, after the first f channel values, were padded with 0.

For example, since the maximum number of frames for the 20% of completion of the grasping movements was 11 (Fig. 3.7), each channel of the movement representation had 11 values. Moreover, if only the fingertips aperture features were engineered, each movement would be represented with 3 channels, one for each feature. Finally, if 9 frames were recorded during this partial movement, the 2 rightmost values of all the channels would be set to 0 (Fig. 3.16).

Following the 1D pseudo-image representation of the movements, each partial movement of the same movement completion interval can be represented in the same way, as an (n_c, n_t) array. For this representation n_c is defined as the number of channels or, equivalently, as the number of kinematic features and n_t is defined as the maximum number of frames of the partial movements. For the previous example, the movement would be represented as an array with shape (3,11).

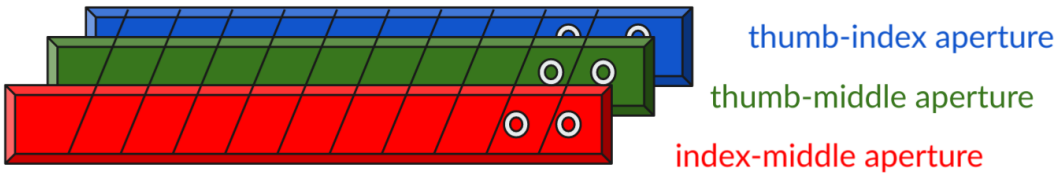

Figure 3.16: The Deep Learning representation of a partial movement (20%) with 9 recorded frames, when only the fingertips aperture features are engineered.

After each instance of the dataset was represented in the same way, an architecture was selected for the DL model (Fig. 3.17). In the first layer of the model, a 1D convolution operation, which was followed by a ReLu activation function, was applied to the (n_c, n_t) input representation, in order to capture feature dependencies in the

temporal domain. In this layer, 24 filters of dimension 7 were used with “same” padding (pad=3). As a result, the dimension of the output of this layer was $(24, n_t)$. Afterwards, a 1D max pooling operation was applied with filter dimension and stride equal to 6, in order to reduce the size of the representation. Consequently, the output of this layer had dimensions $(24, \lfloor \frac{n_t}{6} \rfloor)$. A dropout operation with probability p=0.5 was then applied, before the activation units were flattened. These activation units were fully connected with the three activation units of the output layer. The softmax activation function was used in the final fully-connected layer to extract a probability for each of the small, medium and large objects.

The negative log likelihood loss function and the Adam optimization algorithm were used to perform backpropagation during training. The learning rate of the Adam optimization algorithm was set to 0.001, while the loss function was applied to the logs of the softmax outputs. The model was trained for 500 epochs with a batch size of 16 samples. The PyTorch [58] framework was used.

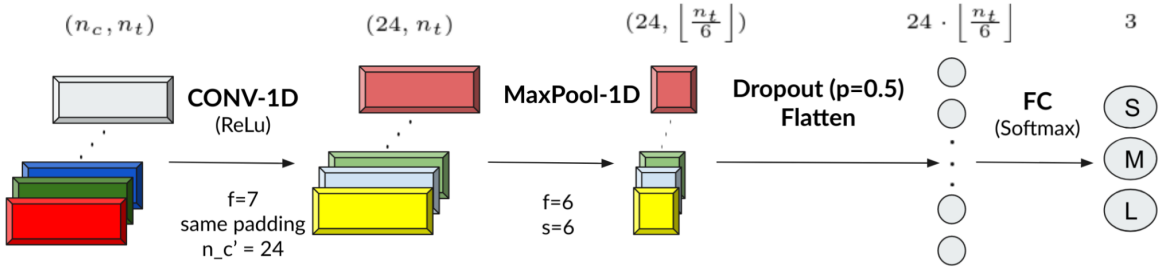

Figure 3.17: The Deep Learning model architecture with the number and the layout of the activation units cited for each layer.

Before the final model architecture was chosen, various other architectures, similar to Fig. 3.17, were evaluated. The most common problem for all of these DL models was high variance; after the model had been trained, the training error was low, but the cross-validation/testing error was high. Similarly, the model’s accuracy rate for the training set movements was high, while the accuracy rate for the cross-validation/testing set movements was low. In general, this is a common phenomenon for DL models and can be attributed to overfitting; the model learns the training set to such a degree that it memorizes noise and, as a result, it cannot generalize to the new data of the cross-validation/testing set. In order to remedy this problem, the final architecture included fewer layers in order to diminish the number of parameters of the model and the dropout regularization technique was used.

3.6. Evaluation Strategies

After the preprocessing of a movement, the frames of the 20%, 40%, 60%, 80% or 100% of the movement’s grasping phase were available. Each model was evaluated for each of these movement completion percentages.

Moreover, in order to evaluate a ML model two strategies, which differed in how the dataset was split into a training and a cross-validation/testing set, were applied. In particular, the following dataset split strategies were used:

- **All-in strategy**

  In the “all-in” evaluation strategy, samples from all participants are used during training. As a result, when a model is used to predict the target object of a movement, it is already trained with other movements of the same participant. In other words, the model can pick up a movement pattern of the specific individual during training and this favours the prediction.

  According to this strategy, the dataset was split into 10 sets. Each set was formed by adding approximately 10% of the movements of each participant-object combination. More specifically, if the movements of a participant-object combination were 30, 3 movements were selected randomly for each of the 10 sets. Since 5 movements were discarded from the dataset due to problematic data collection, a participant-object combination could have less than 30 movements. In this case, the movements of the combination were partitioned randomly to the 10 sets for the latter to be as balanced as possible.

  Following this, a 10-fold strategy was used in order to evaluate a ML model. When the dataset was divided into a training and a cross-validation/testing set according to this strategy, the training set contained approximately 90% of the total movements (approximately 648 movements) and the cross-validation/testing set contained approximately 10% of the total movements (approximately 72 movements).

- **One-out strategy**

  In the “one-out” evaluation strategy, the data of one participant are left out of the training of a model. As a result, a model has to predict which object a participant will grasp, without previously being trained to any of the other movements of this participant. Therefore, a model should be able to generalize based on the movements of the other participants in order to be successful. This is a much more challenging task for a ML model compared to the all-in strategy.

  According to this strategy, the dataset was split into 8 sets. Each set was formed by considering all the movements of a single participant. Afterwards, an 8-fold strategy was followed in order to evaluate each model. When the dataset was divided into a training and a cross-validation/testing set in this way, the training set contained approximately 87.5% of the total movements (approximately 630 movements), while the cross-validation/testing set contained approximately 12.5% of the total movements (approximately 90 movements).

For all the model evaluations for which the same dataset split strategy was used, the same k dataset subsets were used to form the k training set - cross-validation/testing set pairs of the k-fold evaluation scheme.

Finally, in addition to the movement completion percentage and the dataset split strategies, the ML models were evaluated on different feature sets. More specifically, the feature sets of Table 3.1 were taken into consideration.

The first feature set included only the aperture features, while the feature sets 2, 3, 4, 5 and 6 included the aperture features and features only from one of the wrist coordinates, wrist instantaneous speed and wrist coordinates’ dispersion categories. Finally, the feature sets 7 and 8 included a combination of aperture features, wrist

coordinate features and features only from one of the wrist instantaneous speed and wrist coordinates' dispersion categories.

Table 3.1: The feature combinations or feature sets (FS) which were used to evaluate the performance of the models.

	FS 1	FS 2	FS 3	FS 4	FS 5	FS 6	FS 7	FS 8
thumb-index aperture								
thumb-middle aperture								
index-middle aperture								
wrist x-coordinate								
wrist y-coordinate								
wrist x-standard deviation								
wrist y-standard deviation								
wrist xy-standard distance								
wrist x-speed								
wrist y-speed								
wrist xy-speed								

The aperture features were included in all the feature sets, since the feature quality analysis in 3.3.2. revealed observable differences between the values of the aperture features for the small, medium and large object. Furthermore, the feature sets 1, 4 and 6 were chosen to be independent of the spatial configuration of the workspace and data collection set-up than the rest of the feature sets (2,3,5,7,8), since the latter included wrist kinematic features computed for the x- and y- axes instead of the xy-plane. None of the feature sets included similar kinematic feature information both in the x- and y-axes and in the xy-plane (e.g. wrist x-speed, wrist y-speed and wrist xy-speed), to avoid information recurrence. Similarly, none of the feature sets included both wrist instantaneous speed and wrist coordinates' dispersion features, since each of these features was used to measure the speed of the wrist.

4. RESULTS

4.1. Models' Evaluations and Comparisons

In this section, the performance of the "traditional" ML and DL methods is presented following the Methodology presented in Chapter 3. The section is divided further into two subsections:

- **"All-in" Results (4.1.1)**
- **"One-out" Results (4.1.2)**

The subsections present the evaluation of the models based on the corresponding dataset split strategy. In both subsections, the ML models are first compared to each other for all the different feature sets, and afterwards, for each feature set, the DL model is compared to the corresponding top-performing ML model. For each feature set, in the ML models evaluations, the confusion matrices of the top-performing ML model are plotted for each of the 20%, 40%, 60%, 80% and 100% movement completion percentages. If for a given feature set there was not a ML model superior from the others for all the movement completion percentages, one of the top-performing ML models was arbitrarily chosen both to plot the confusion matrices and to compare to the DL model. Similarly for each feature set, in the DL model evaluations, the confusion matrices of the model were plotted.

The confusion matrix (CM) of a model was calculated as the average of the k confusion matrices, which were computed for the model in the k-fold evaluation scheme. The standard deviation of the cell values of the confusion matrix was also calculated and it is cited in parentheses in the corresponding cell. In the confusion matrix C, the $C_{i,j}$ value corresponds to the number of observations known to be in group i and predicted to be in group j. The confusion matrix was normalized over the true conditions (rows) and, therefore, the confusion matrix of a perfect model would be the identity matrix.

The evaluation results of the models are listed in Tables 4.1-4.32. In each table cell, the average and the standard deviation of the k accuracy rates, which were computed in the k-fold evaluation scheme, are cited. The models were compared based primarily on the average accuracy rate for each movement completion percentage, while the standard deviations were used to identify whether the model's performance was consistent over the k different dataset splits. In case two models had the same average accuracy rates, the model with the lower standard deviation would be considered as the superior one, since its performance would be more consistent for the entire dataset.

4.1.1. "All-in" Results

The results of the "all-in" strategy for the "traditional" ML models and the DL model are presented in subsections 4.1.1.1 and 4.1.1.2 respectively. At the end of each subsection, the results are summarized and commented on.

4.1.1.1. "Traditional" Machine Learning Results

The results of the ML models for the **1st feature set** are presented in Table 4.1 and Figure 4.1. The 1st feature set includes only the aperture features (Table 3.1). The top-performing ML models for this feature set were the Random Forest and **Extra Trees** models. During the early stages of the movements (20%, 40%), the accuracy rates of the top-performing models were above 55% and 70% respectively. For the 60%, 80%

and 100% of the movements, the accuracy further increased to approximately 80%, 85% and 90% respectively. The Extra Trees model was selected as the top-performing ML model and its CMs were plotted in Fig. 4.2. The CMs shows that the majority of the small objects were identified correctly from the beginning of the movement and the predictions improved over time to the extent that almost all the small-labeled full movements were correctly classified. The medium-labeled movements were the most difficult to classify at the beginning of the movements. Finally, at the late stages of the movements, some medium objects were misclassified as large and vice-versa.

Table 4.1: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 1st feature set.

	20%	40%	60%	80%	100%
RF	59.01 (4.56)	73.29 (3.21)	79.29 (3.88)	85.59 (2.97)	89.37 (3.60)
GB	55.10 (3.65)	72.85 (5.13)	78.45 (3.76)	84.88 (3.51)	90.63 (2.27)
ET	**57.75 (3.54)**	**73.14 (3.96)**	**80.54 (3.30)**	**86.84 (3.71)**	**92.03 (2.41)**
SVM	52.57 (4.86)	65.33 (4.43)	71.74 (4.38)	75.94 (4.33)	80.27 (2.66)
GP	50.75 (5.36)	61.84 (4.90)	75.24 (4.21)	79.71 (3.03)	79.88 (3.97)

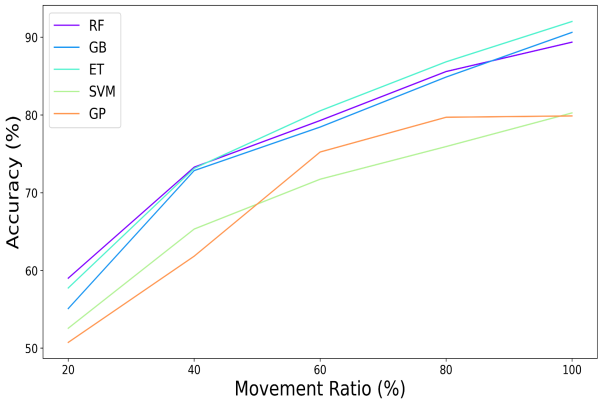

Figure 4.1: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 1st feature set.

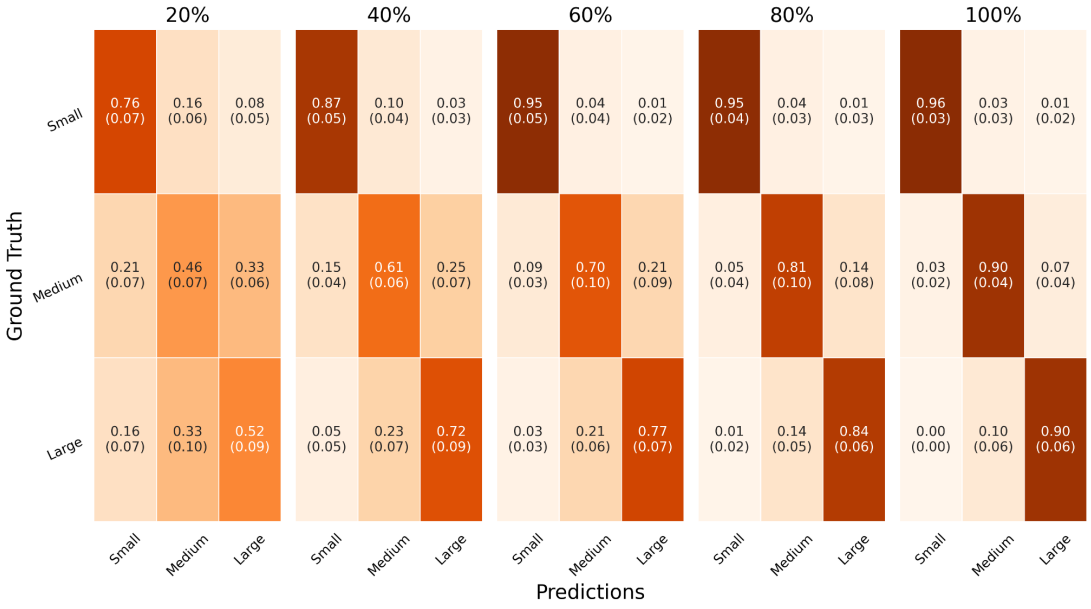

Figure 4.2: The confusion matrices of the Extra Trees ML model for the "all-in" dataset split and the 1st feature set with respect to the movement completion percentages.

The results of the ML models for the **2nd feature set** are presented in Table 4.2 and Fig. 4.3. This feature set contains the aperture and the wrist coordinates features (Table 3.1). The top-performing ML model was **Extra Trees** for all the movement completion percentages. Moreover, based on the standard deviations and compared to the other

ML models, the predictions of Extra Trees were more consistent across different dataset splits. For this model, the accuracy rate was over 65% at 20%, over 80% at 40%, over 85% at 60% and, finally, over 90% at 80% and 100%. Based on Fig. 4.4, after 60% of the movements, the small objects were correctly discerned with almost 100% accuracy rate by the Extra Trees model. Moreover, while the predictions for the medium and large objects were approximately 50% and 65% accurate respectively at 20%, they gradually improved to over 90% for the full movements.

Table 4.2: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 2nd feature set.

	20%	40%	60%	80%	100%
RF	61.66 (4.45)	78.04 (3.23)	84.88 (4.72)	88.66 (3.69)	90.34 (3.23)
GB	57.63 (4.11)	75.10 (3.38)	83.77 (2.90)	87.69 (3.58)	91.46 (3.64)
ET	**66.27 (5.51)**	**80.14 (2.91)**	**86.01 (2.84)**	**91.88 (2.64)**	**93.84 (2.02)**
SVM	49.10 (2.79)	58.61 (4.78)	61.80 (4.77)	66.98 (2.80)	72.03 (3.14)
GP	41.54 (4.84)	50.49 (4.37)	56.51 (6.28)	61.83 (4.34)	63.08 (3.02)

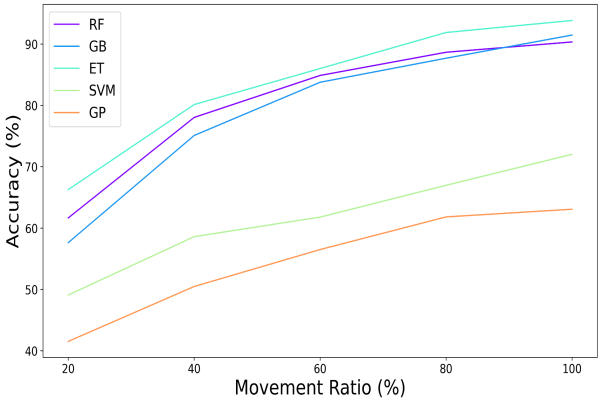

Figure 4.3: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 2nd feature set.

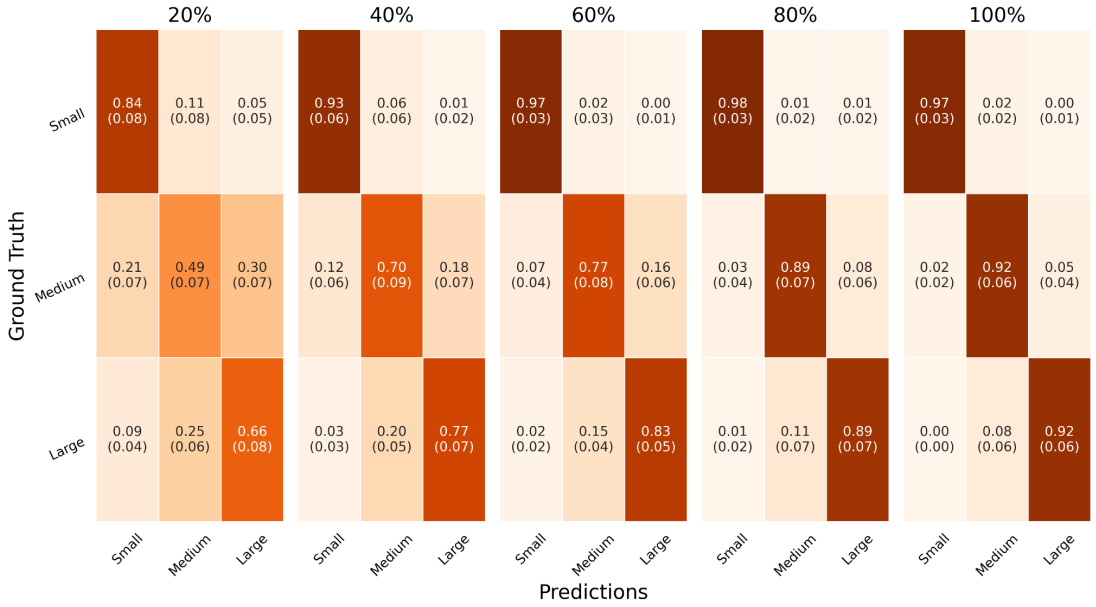

Figure 4.4: The confusion matrices of the Extra Trees ML model for the "all-in" dataset split and the 2nd feature set with respect to the movement completion percentages.

The ML models were evaluated for the **3rd feature set** and the results are presented in Table 4.3 and Fig. 4.5. This feature set includes the aperture and the wrist coordinates' dispersion axes features (Table 3.1). The top-performing models were **Extra Trees** and Gradient Boosting and the former was selected for Fig. 4.6 and the DL comparison. The Extra Trees model achieved accuracy rates over 60% at 20%, over 75% at 40%, over

80% at 60% and 80% and over 90% at 100%. Based on Fig. 4.6, the small objects were successfully predicted with over 80% at 20%, over 90% at 40% and over 95% from 60% and afterwards. The most difficult movements to be classified were the medium-labeled ones, especially at the beginning (20%), since they were misclassified both as small or large. Over time, the accuracy of the predictions improved, as the medium and large objects were almost never misclassified as small for the full movements, while they were seldomly misclassified as each other.

Table 4.3: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 3rd feature set.

	20%	40%	60%	80%	100%
RF	59.15 (5.30)	73.57 (3.89)	81.66 (3.30)	86.57 (2.69)	89.37 (2.10)
GB	55.09 (5.21)	73.01 (3.77)	79.99 (4.67)	86.28 (3.92)	90.77 (2.74)
ET	**63.36 (3.62)**	**77.19 (4.27)**	**83.06 (2.47)**	**87.41 (2.37)**	**90.62 (3.16)**
SVM	53.13 (3.22)	69.24 (4.62)	73.28 (4.52)	75.24 (4.30)	80.14 (2.81)
GP	46.17 (4.51)	63.64 (1.89)	73.98 (4.23)	77.06 (2.64)	79.18 (3.99)

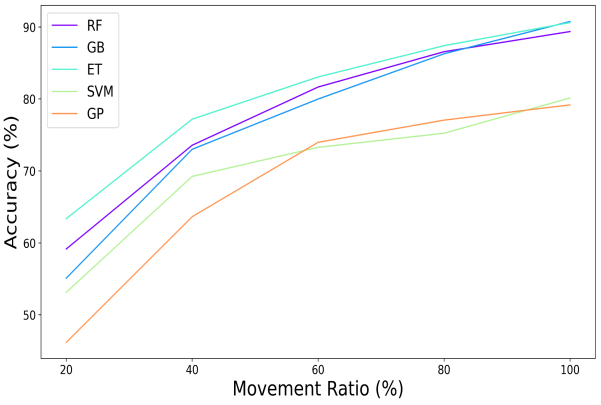

Figure 4.5: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 3rd feature set.

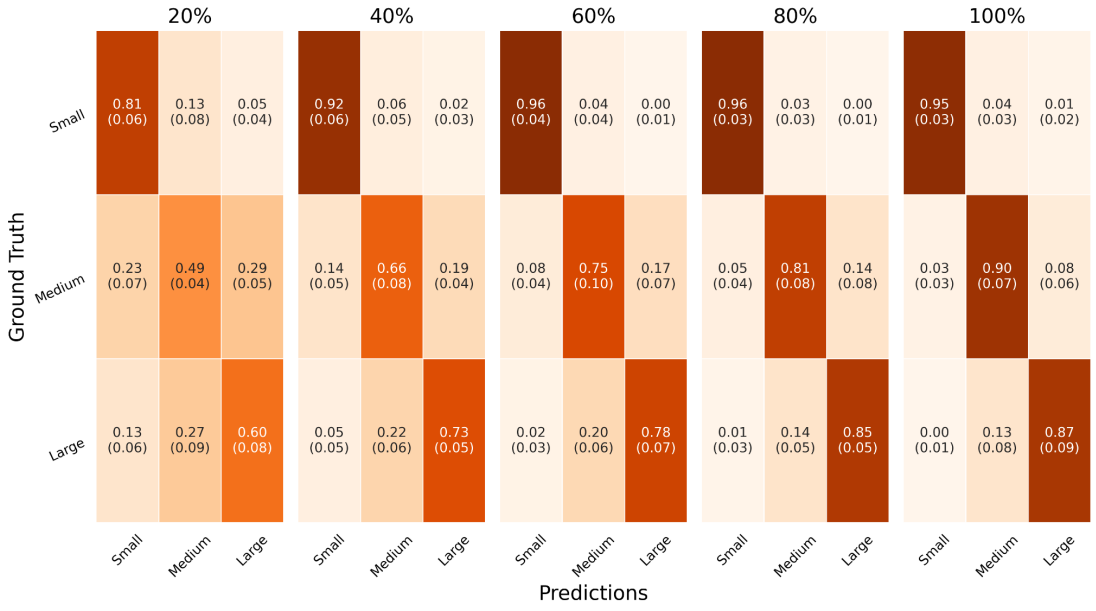

Figure 4.6: The confusion matrices of the Extra Trees ML model for the "all-in" dataset split and the 3rd feature set with respect to the movement completion percentages.

The results of the ML models for the **4th feature set** are presented in Table 4.4 and Fig. 4.7. The 4th feature set includes the aperture and the wrist coordinates' dispersion plane features (Table 3.1). The best-performing models were **Extra Trees**, which was selected for the comparisons, and Random Forest. The accuracy rate for the Extra Trees model was approximately 60% at 20% and gradually improved to over 90% for

the full movements. Based on Fig. 4.8, the small object was easily discerned from the medium and the large one, even from the beginning. From 40% of the movements and onwards, the small objects were predicted successfully with an accurate rate of approximately 90% or higher. The large objects, which were incorrectly classified as small, were approximately 10% at 20% and approximately 2% or less from 60% and afterwards. The main reason for the accuracy drop of the model was that medium- and large- labeled movements were misclassified as each other.

Table 4.4: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 4th feature set.

	20%	40%	60%	80%	100%
RF	61.39 (3.66)	77.76 (4.11)	81.65 (4.59)	86.00 (3.65)	89.65 (3.14)
GB	56.77 (3.40)	74.83 (5.03)	80.41 (4.42)	85.31 (2.77)	90.07 (2.92)
ET	**61.81 (4.81)**	**77.19 (3.67)**	**83.20 (3.92)**	**87.54 (2.16)**	**91.32 (2.44)**
SVM	53.83 (3.27)	68.41 (4.69)	72.16 (3.46)	74.68 (3.56)	79.99 (2.85)
GP	47.53 (5.98)	64.49 (3.37)	73.85 (4.33)	77.62 (2.47)	78.89 (3.13)

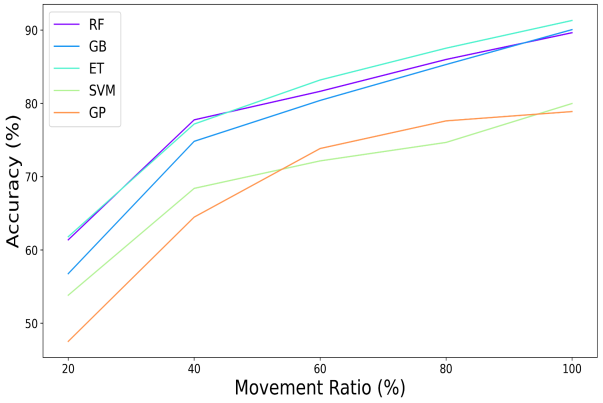

Figure 4.7: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 4th feature set.

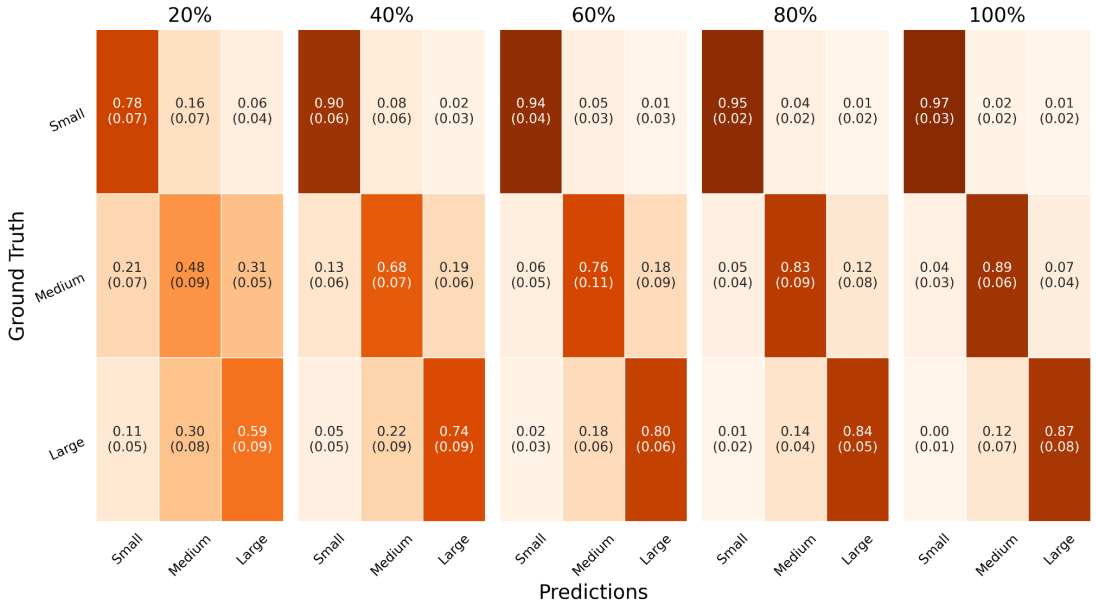

Figure 4.8: The confusion matrices of the Extra Trees ML model for the "all-in" dataset split and the 4th feature set with respect to the movement completion percentages.

For the **5th feature set** the results are presented in Table 4.5 and Fig. 4.9. This feature set contains the aperture and the wrist speed axes features (Table 3.1). The best-performing ML models were **Random Forest**, which was selected for further comparisons, and Extra Trees. On the other hand, the Gaussian Process model performed as poorly as random guessing. The accurate rate of the Random Forest

model was approximately 60% at 20% and above 70%, 80%, 85% and 90% at 40%,60%,80% and 100% of the movement respectively. Based on Fig. 4.10, the small object was correctly detected with accuracy above 75% at 20% and approximately 90% or more at 40% and onwards. The medium-labeled movements were the most difficult to classify at the beginning of the movement (approximately 40% accuracy), but the performance gradually improved to approximately 90% for the full movement. From 60% of the movement and afterwards, approximately only 1% of the large-labeled movements were misclassified as small, while 10% to 20% were misclassified as medium.

Table 4.5: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 5th feature set.

	20%	40%	60%	80%	100%
RF	**59.15 (5.52)**	**73.43 (4.05)**	**80.40 (3.96)**	**85.45 (2.10)**	**90.35 (3.78)**
GB	54.82 (3.97)	72.58 (3.39)	78.17 (3.81)	84.33 (2.72)	89.66 (3.66)
ET	58.03 (3.52)	73.56 (5.21)	80.68 (3.62)	86.30 (1.48)	89.93 (2.81)
SVM	33.71 (3.94)	42.79 (3.74)	54.67 (5.29)	62.93 (4.67)	59.73 (2.85)
GP	33.29 (0.49)	33.29 (0.49)	33.29 (0.49)	33.29 (0.49)	33.57 (0.84)

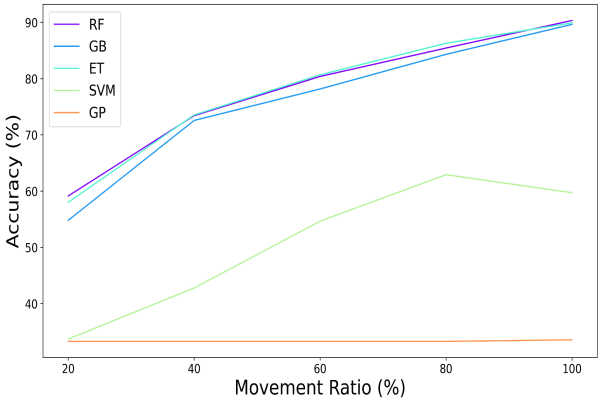

Figure 4.9: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 5th feature set.

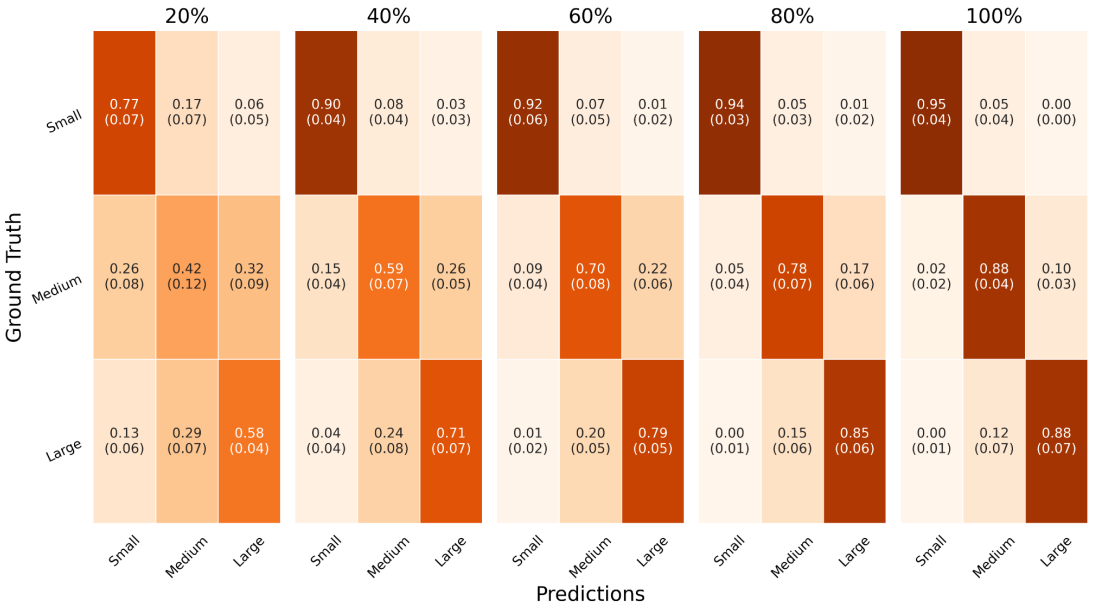

Figure 4.10: The confusion matrices of the Random Forest ML model for the "all-in" dataset split and the 5th feature set with respect to the movement completion percentages.

The results for the **6th feature set** are presented in Table 4.6 and Fig. 4.11. The 6th feature set includes the aperture and the wrist speed plane features (Table 3.1). The top-performing ML models were Random Forest and **Extra Trees.** The latter was further investigated. The Gaussian Process model's performance was almost identical to

random guessing. The Extra Tree model's accuracy rate was below 60% at 20%, over 70% at 40%, below 80% at 60%, over 85% at 80% and, finally, below 90% at 100% of the movements. Based on Fig. 4.12, the medium objects were the most difficult to detect with their accuracy rates ranging approximately from 40% for 20% of the movement to 85% for the full movement. On the other hand, the small objects were easily detected (their accuracy rates were approximately 80% at 20%, 90% at 40% and over 95% after 60%). The large objects' accuracy rates ranged from 55% to 85%, with the main reason for the suboptimal accuracy being the misclassification as medium (15% to 30% of the large movements were misclassified as medium at 100% and 20% respectively).

Table 4.6: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 6th feature set.

	20%	40%	60%	80%	100%
RF	58.03 (3.55)	74.12 (3.09)	79.15 (4.33)	85.59 (3.85)	90.07 (3.78)
GB	53.43 (4.03)	73.15 (4.88)	79.58 (3.35)	84.32 (3.81)	89.37 (2.74)
ET	**59.01 (4.36)**	**72.87 (5.46)**	**79.85 (3.16)**	**86.56 (3.87)**	**89.08 (4.17)**
SVM	34.55 (3.08)	41.94 (5.31)	53.97 (4.90)	61.93 (5.36)	62.81 (5.31)
GP	33.29 (0.49)	33.29 (0.49)	33.57 (0.67)	33.29 (0.49)	33.98 (1.20)

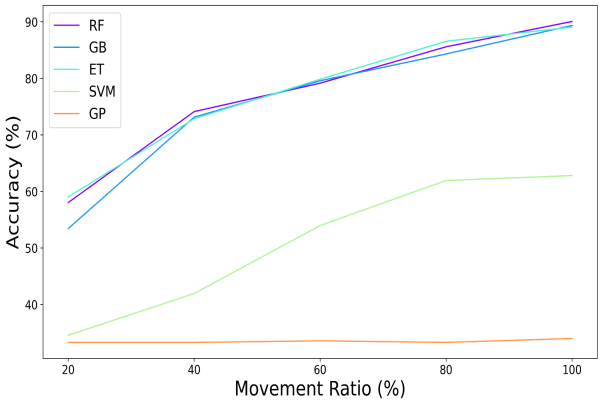

Figure 4.11: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 6th feature set.

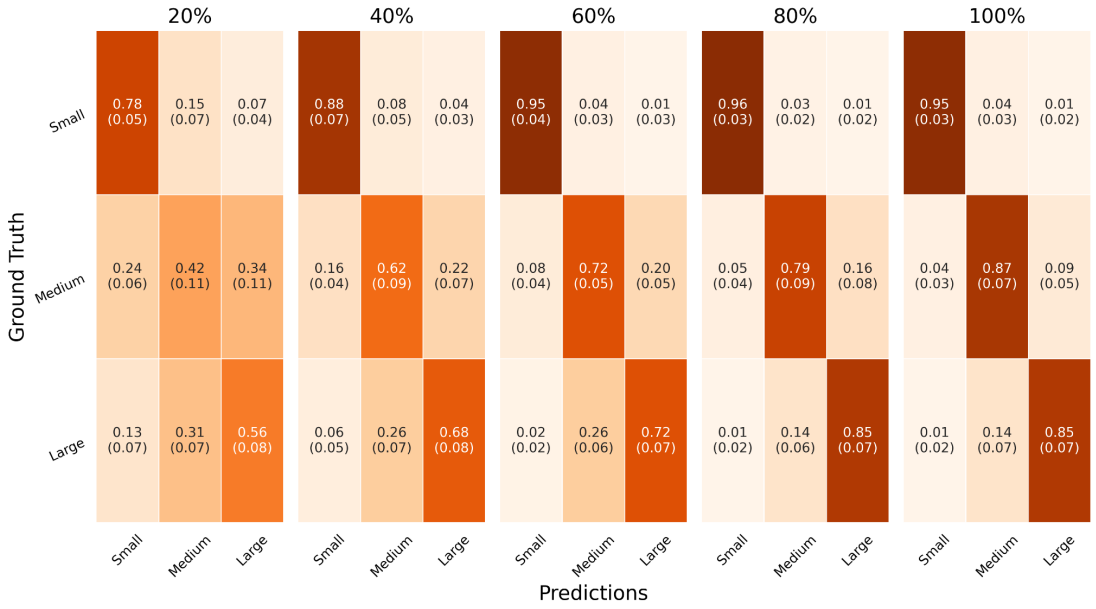

Figure 4.12: The confusion matrices of the Extra Trees ML model for the "all-in" dataset split and the 6th feature set with respect to the movement completion percentages.

For the **7th feature set** the results of the ML models are presented in Table 4.7 and Fig. 4.13. The feature set includes the aperture, the wrist coordinate and the wrist coordinates' dispersion axes features (Table 3.1). The top-performing model was **Extra**

Trees with an accurate rate below 65%, over 80%, over 85%, below 90% and over 90% at 20%, 40%, 60%, 80% and 100% of the movements respectively. Based on Fig 4.14, the small-labeled movements were classified correctly even from the beginning of the movement, since only 20%, 7% and 3% of the small objects were misclassified at 20%, 40% and after 60% respectively. The medium objects were the most difficult to classify, in general, with their accuracy rates ranging from below 50% at 20% to above 90% for the full movement. The large objects' accuracy rate at 20% was 65% and gradually improved to 90% at 100%, while only below 1% of the large objects were misclassified as small after 60%.

Table 4.7: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 7th feature set.

	20%	40%	60%	80%	100%
RF	63.21 (4.53)	76.49 (5.62)	84.33 (3.88)	88.25 (1.91)	90.35 (2.84)
GB	57.47 (6.28)	75.94 (4.15)	82.65 (3.71)	87.96 (3.17)	90.76 (4.02)
ET	**64.90 (3.61)**	**80.41 (3.49)**	**86.28 (2.79)**	**89.37 (3.08)**	**92.86 (2.85)**
SVM	49.24 (2.83)	58.47 (5.09)	61.80 (4.81)	66.69 (3.89)	71.33 (3.33)
GP	37.49 (2.32)	48.25 (4.08)	53.01 (4.21)	58.61 (5.19)	61.95 (3.51)

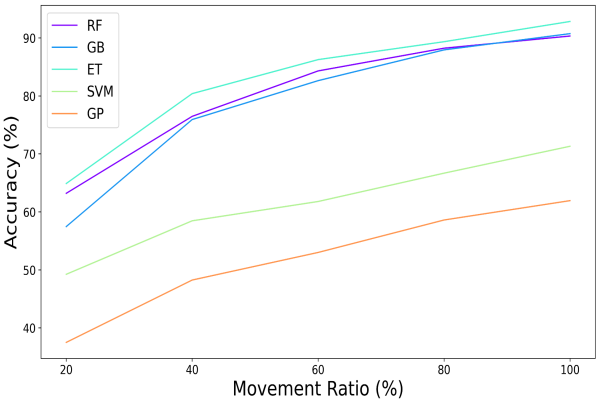

Figure 4.13: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 7th feature set.

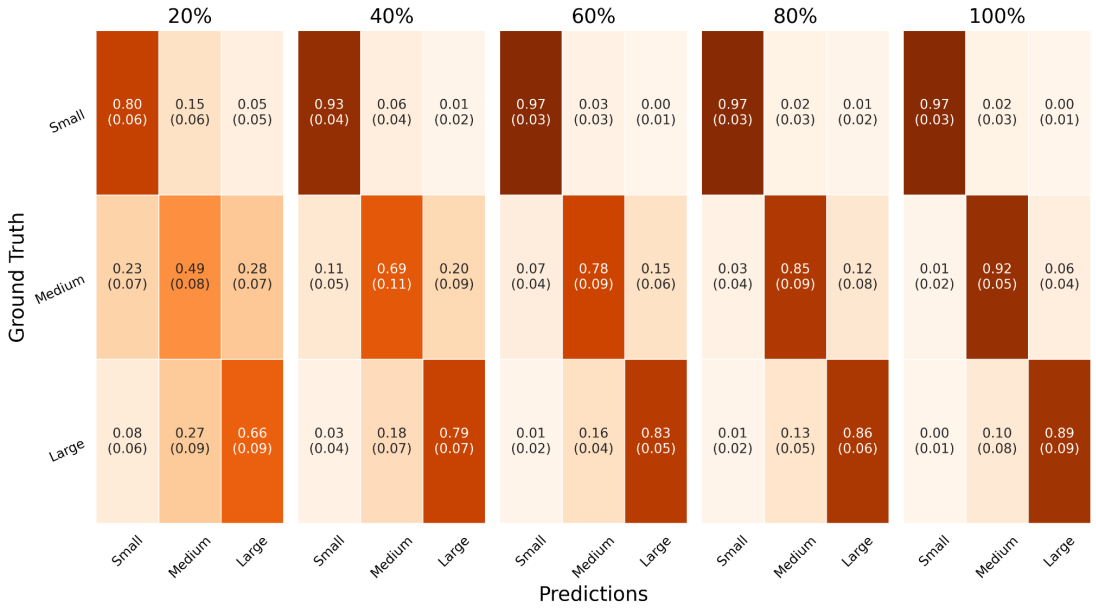

Figure 4.14: The confusion matrices of the Extra Trees ML model for the "all-in" dataset split and the 7th feature set with respect to the movement completion percentages.

The results regarding the **8th feature set** are presented in Table 4.8 and Fig. 4.15. The 8th feature set consists of the aperture, the wrist coordinate and the wrist speed axes features (Table 3.1). The top-performing ML model was **Extra Trees** for each of the

movement completion intervals. The model' accuracy rate was above 60% at 20% and gradually increased by approximately 30% for the full movement. Based on Fig. 4.16, the Extra Trees model classified the small-labeled movements with over 80% and 90% accuracy rate at the beginning of the movement (20%, 40% of the movement respectively), while the accuracy increased to over 95% after 60%. The large-labeled movements were easily discerned from the small-labeled movements at every movement competition interval with the error below 10%, 3% and 1% at 20%, 40% and after 60% respectively. Similarly, the medium object was not misclassified as small at the later stages of the movements, since the error decreased from above 20% at 20% to below 10% at 60% and below 5% after 80%. The primary cause for the suboptimal performance of the model was that over 10% of the medium and large objects were misclassified as each other for every interval other than 100%, at which the error for the medium objects, which were predicted to be large, was approximately 5%.

Table 4.8: The evaluation of the "traditional" ML models for the "all-in" dataset split and the 8th feature set.

	20%	40%	60%	80%	100%
RF	61.39 (3.34)	76.78 (2.68)	83.62 (3.60)	87.40 (3.52)	90.07 (3.05)
GB	57.06 (2.19)	73.85 (2.99)	83.07 (3.75)	87.41 (3.44)	90.20 (4.20)
ET	**63.64 (4.90)**	**79.43 (3.41)**	**85.99 (5.14)**	**88.81 (2.15)**	**93.00 (1.78)**
SVM	33.57 (4.41)	42.50 (5.40)	53.41 (4.60)	58.17 (4.84)	52.19 (4.68)
GP	33.29 (0.49)	33.29 (0.49)	33.29 (0.49)	33.29 (0.49)	33.29 (0.49)

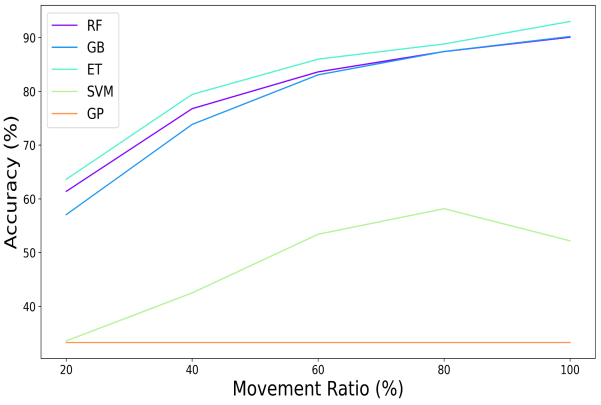

Figure 4.15: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "all-in" dataset split and the 8th feature set.

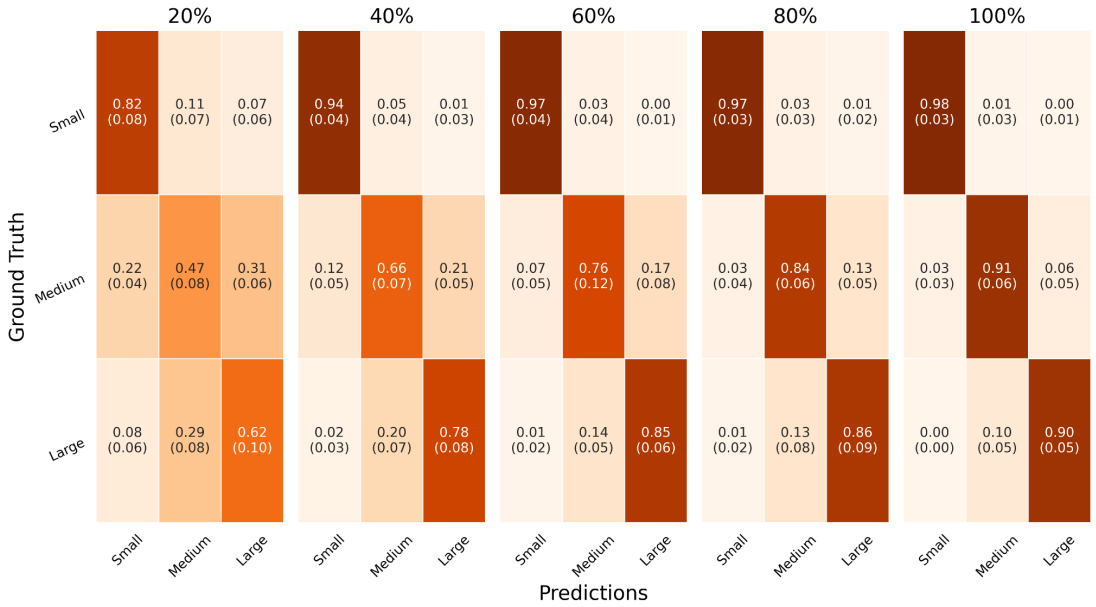

Figure 4.16: The confusion matrices of the Extra Trees ML model for the "all-in" dataset split and the 8th feature set with respect to the movement completion percentages.

In summary, the *Extra Trees*, *Random Forest* and *Gradient Boosting* ML models outperformed the Support Vector Machine and Gaussian Process models by a large margin regardless of the feature set. In general, the accuracy rates of the three top-performing models were consistent across different feature sets. Specifically, the accuracy rates ranged approximately from 60% at 20% to 90% at 100%. Moreover, the standard deviations of the three top-performing ML models were around 5% or less for each movement completion interval. This indicated that the models achieved a good performance for the entire dataset, instead of a subset of the data.

The **Extra Trees** model logged the highest accuracy rates among the three top-performing models for most feature sets. For the **2nd feature set**, which included the aperture and the wrist coordinate features, the Extra Tree's accuracy rates were 66.27%, 80.14%, 86.01%, 91.88% and 93.84% at 20%, 40%, 60%, 80% and 100% respectively. The Extra Trees's accuracy rates for the workspace-independent **4th feature set**, which included the aperture and the wrist coordinates' dispersion plane features, were 61.81%, 77.19%, 83.20%, 87.54% and 91.32% at 20%, 40%, 60%, 80% and 100% respectively.

Moreover, similar conclusions can be drawn based on all the CMs of the top-performing ML models. It can be observed that the accuracy rates gradually improved for each 20% movement completion intervals. *The ML models could easily discern the small-labeled movements from the medium- and large-labeled movements even from the beginning. Similarly, the large-labeled movements were not misclassified as small even for the 20% movement completion interval. On the other hand, the large objects were falsely predicted to be medium more often. At the same time, the medium objects were initially misclassified as both small and large objects, but for the later stages of the movements, the models learned to identify the medium objects as not small.*

4.1.1.2. Deep Learning Results

For the **1st feature set** the Neural Net DL model was evaluated and its results are presented in Table 4.9 and Fig. 4.17. The 1st feature set contains the fingertip aperture features (Table 3.1). The model's accuracy increased from approximately 50% at 20% to over 80% for the full movement. Compared to the top-performing Extra Trees ML model, the DL model achieves lower accuracy rates for each of the movement completion intervals. Furthermore, based on the models' standard deviations, the ML model achieves more consistent accuracy rates throughout the entire dataset than the DL model.

Based on Fig. 4.18, at the early stages of the movements (20%, 40%), the small-labeled movements were identified correctly with accuracy rates over 65% and 75%. For later stages of the movements, the accuracy rates increased to over 90%. The medium-labeled movements were initially misclassified as both small and large, but after 80% of the movement the model learned to identify a medium object as non-small with error rate below 5%. Similarly, the large-labeled movements were misclassified as both small and medium at 20%, but at the 60% movement completion interval and onwards, the model learned to identify a large object as non-small with error rate below 2.5%.

Table 4.9: The results of the Neural Net DL model compared to the Extra Trees ML model for the "all-in" dataset split and the 1st feature set.

	20%	40%	60%	80%	100%
NN	50.76 (4.35)	60.56 (5.86)	71.59 (5.86)	75.52 (5.44)	83.22 (4.80)
ET	**57.75 (3.54)**	**73.14 (3.96)**	**80.54 (3.30)**	**86.84 (3.71)**	**92.03 (2.41)**

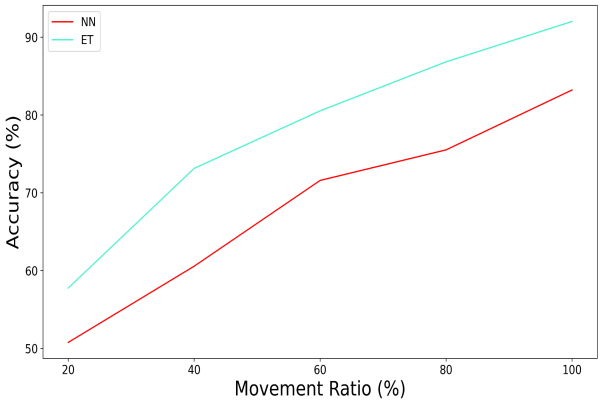

Figure 4.17: The accuracy rates of the Neural Net DL model and of the Extra Trees ML model with respect to the movement ratio for the "all-in"dataset split and the 1st feature set.

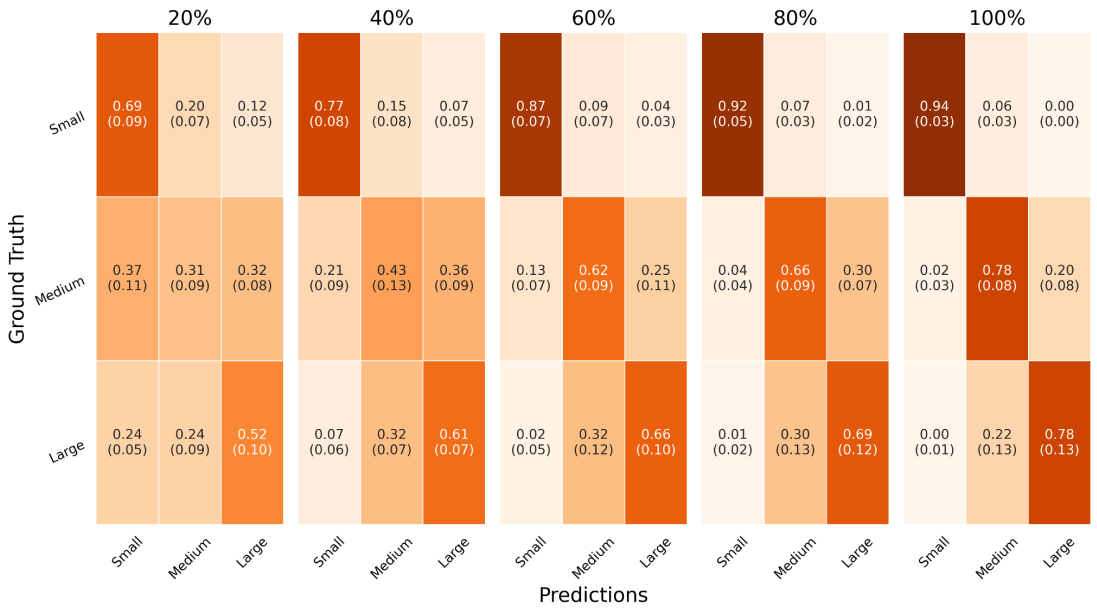

Figure 4.18: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 1st feature set with respect to the movement completion percentages.

For the **2nd feature set** the results of the DL model are presented in Table 4.10 and Fig. 4.19. The feature set contains the aperture features and the wrist coordinates (Table 3.1). The accuracy rates of the DL model ranged from 40% at 20% to 70% for the full movement. Moreover, the ML model achieved a more consistent performance than the DL model through the entire dataset, as it is evident by the standard deviations.

Based on Fig. 4.20, one can observe that at the early stages, the majority of the movements were classified as large independently of the target label (over 45% small, over 60% medium and over 75% large objects were classified as large at 20%). The accuracy rate for the small object increased from approximately 40% to 90% throughout the movement. The DL model did not perform well when classifying the medium objects, as until 80% the model performed worse or the same with random guessing. Finally, the accuracy rate for the large object was approximately 80% until 60% of the movement, but afterwards, the accuracy dropped to 65% for 80% and 100%, since a large

percentage of the large-labeled movements were misclassified as medium.

Table 4.10: The results of the Neural Net DL model compared to the Extra Trees ML model for the "all-in" dataset split and the 2nd feature set.

	20%	40%	60%	80%	100%
NN	42.79 (2.19)	53.72 (4.17)	61.11 (3.71)	66.97 (7.27)	69.93 (4.52)
ET	**66.27 (5.51)**	**80.14 (2.91)**	**86.01 (2.84)**	**91.88 (2.64)**	**93.84 (2.02)**

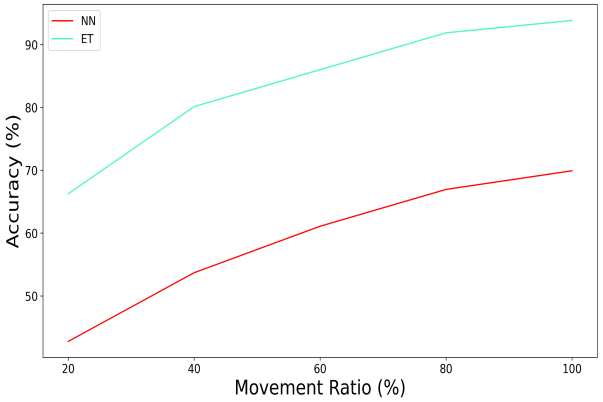

Figure 4.19: The accuracy rate of the Neural Net DL model and of the Extra Trees ML model with respect to the movement ratio for the "all-in"dataset split and the 2nd feature set.

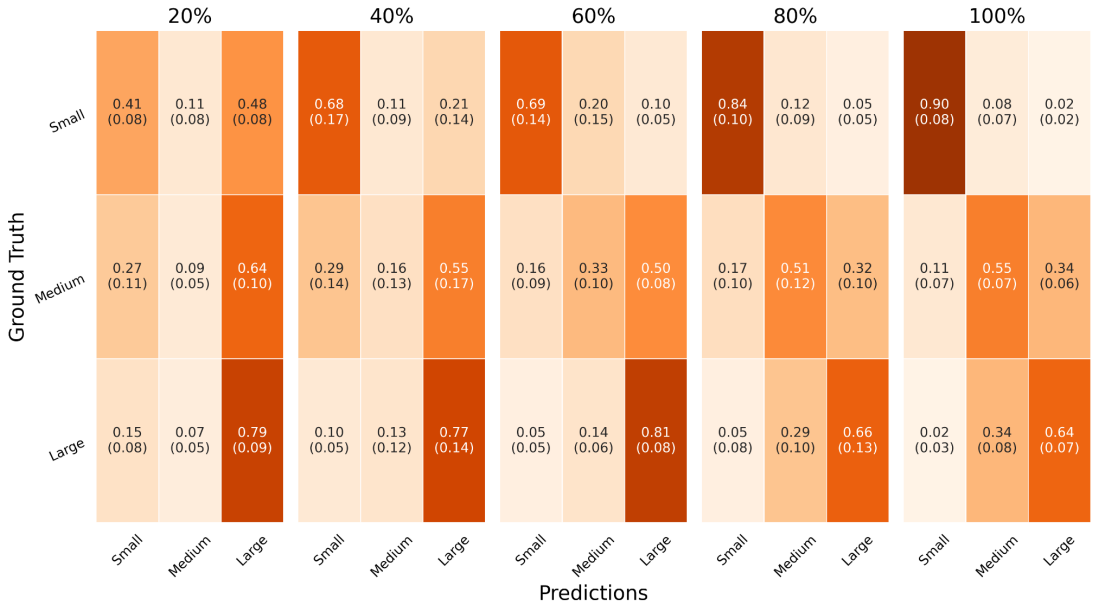

Figure 4.20: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 2nd feature set with respect to the movement completion percentages.

For the **3rd feature set** the results are presented in Table 4.11 and Fig. 4.21. This feature set includes the aperture and the wrist coordinates' dispersion axes features (Table 3.1). Specifically, the accuracy rate of the DL model was above 50%, 60%, 75%, 80% and 85% at 20%, 40%, 60%, 80% and 100% of the movement respectively. Compared to the top-performing Extra-Trees ML model, the DL model's accurate rates are lower than the ML model's corresponding ones by approximately 5% to 10%. Moreover, at 20%, 60% and 80% of the movement, the standard deviations of the DL model were significantly higher than those of the ML model. Therefore, for these movement completion intervals, the DL model did not perform equally successfully for the whole dataset, as opposed to the ML model.

Based on Fig. 4.22, the model's performance for the medium-labeled movements at 20% was approximately equivalent with random guessing. However, the model's

accuracy rate for the small object gradually increased to over 80% for the full movement. The accuracy rate of the model for the small object was over 65% and approximately 75% for the early stages of the movement (20%, 40% respectively) and increased to approximately 90% or more after 60%. Finally, the large object was misclassified as both small and medium at 20%, but after 60% the model learned to identify the large object as non-small with an error rate below 1%.

Table 4.11: The results of the Neural Net DL model compared to the Extra Trees ML model for the "all-in" dataset split and the 3rd feature set.

	20%	40%	60%	80%	100%
NN	51.21 (6.45)	63.08 (3.25)	75.25 (5.29)	81.09 (6.68)	86.99 (2.80)
ET	**63.36 (3.62)**	**77.19 (4.27)**	**83.06 (2.47)**	**87.41 (2.37)**	**90.62 (3.16)**

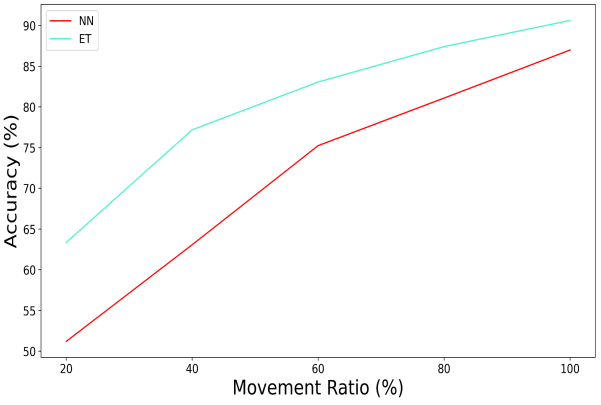

Figure 4.21: The accuracy rate of the Neural Net DL model and of the Extra Trees ML model with respect to the movement ratio for the "all-in"dataset split and the 3rd feature set.

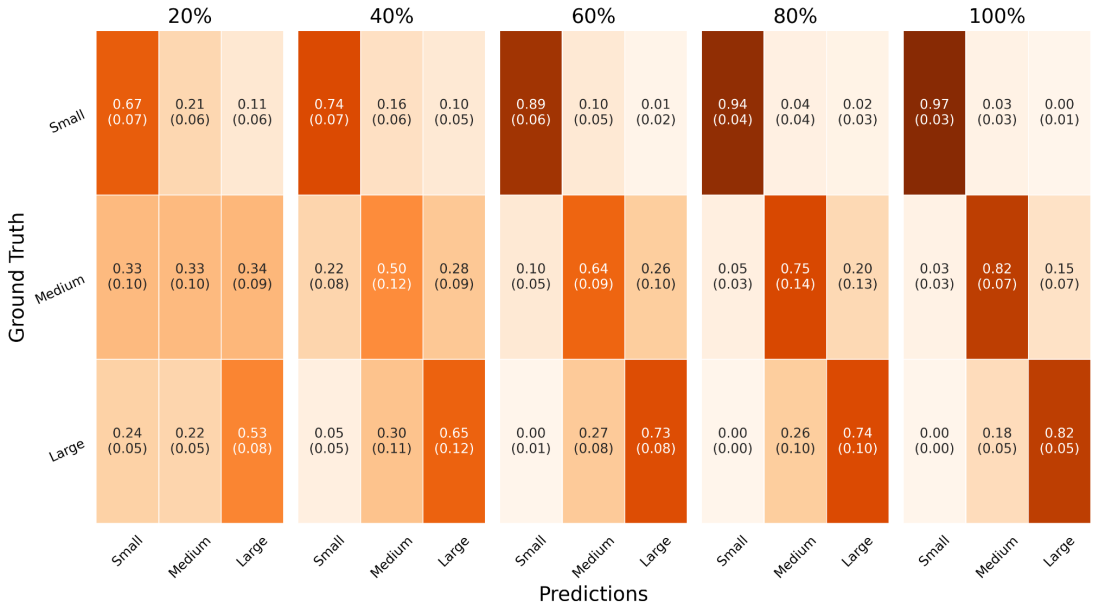

Figure 4.22: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 3rd feature set with respect to the movement completion percentages.

The results of the DL model for the **4th feature set** are presented in Table 4.12 and Fig. 4.23. This feature set contains the aperture and the wrist coordinates' dispersion plane features (Table 3.1). The model's accuracy rate gradually increased from approximately 50% for 20% of the movement to over 80% for the whole grasping movement. Compared to the top-performing Extra Trees ML model, the DL model was outperformed for all the movement completion intervals by approximately 10%.

Based on Fig. 4.24, the model could identify the small objects more easily than the other objects for all the movement completion intervals. Specifically, the accuracy of the small-labeled movements was around 70% at 20% and increased by approximately 25% for the full movement. Regarding the medium-labeled movements, the model performed worse than random guessing at 20%, but the accuracy rate increased to over 70% at 80% and 100%. For these late stages of the movement, the model identified the medium objects as non-small with error rate below 5%. Finally, the model identified the large-labeled movements as non-small after 60% with a lower error rate than 1%.

Table 4.12: The results of the Neural Net DL model compared to the Extra Trees ML model for the "all-in" dataset split and the 4th feature set.

	20%	40%	60%	80%	100%
NN	49.37 (5.64)	63.91 (4.48)	74.41 (5.74)	80.69 (3.94)	83.36 (1.92)
ET	**61.81 (4.81)**	**77.19 (3.67)**	**83.20 (3.92)**	**87.54 (2.16)**	**91.32 (2.44)**

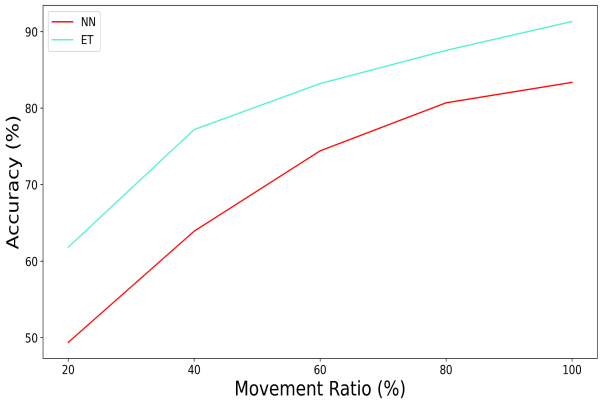

Figure 4.23: The accuracy rate of the Neural Net DL model and of the Extra Trees ML model with respect to the movement ratio for the "all-in"dataset split and the 4th feature set.

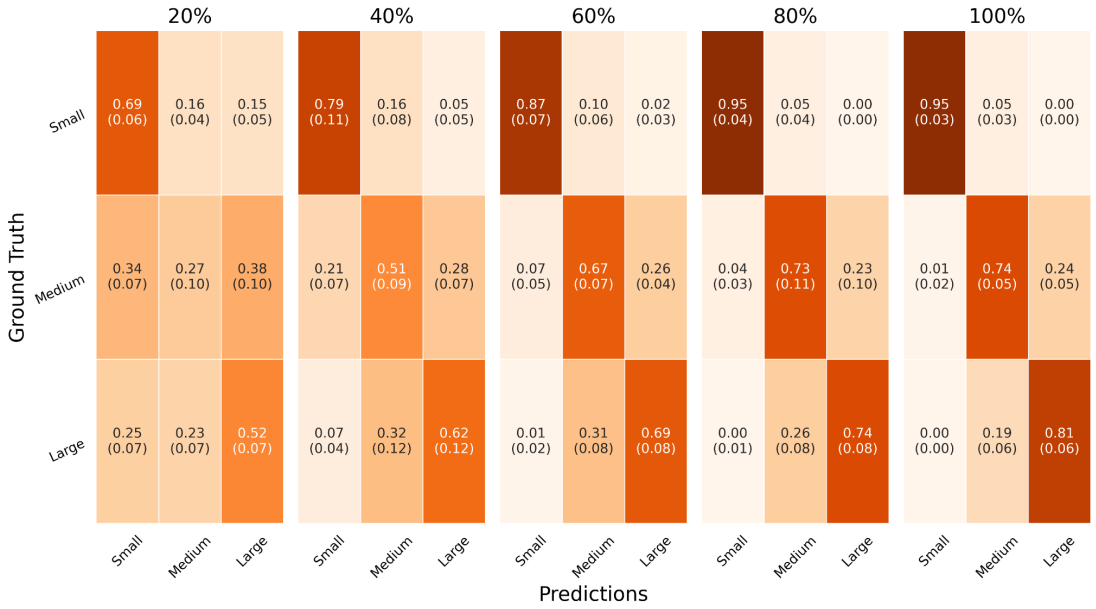

Figure 4.24: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 4th feature set with respect to the movement completion percentages.

The results of the Neural Net DL model for the **5th feature set** are presented in Table 4.13 and Fig. 4.25. The feature set consists of the aperture and the wrist speed axes features (Table 3.1). The DL model's accuracy rates ranged from over 35% to below 50%. Therefore, the model was outperformed by a large margin in comparison with the top-performing Random Forest model. Specifically, the Random Forest model's

accuracy rates were higher than the Neural Net's accuracy rates by at least 20% for each movement completion interval. Moreover, for each interval the standard deviations of the Neural Net were higher than the Random Forest's corresponding values, which indicated that the DL model did not perform equally well throughout the whole dataset.

Finally, based on Fig. 4.26, the medium- and the large-labeled movements had accuracy rates less than 40% and 50% respectively for each movement completion interval. The model achieved its best results for the small object, with accurate rates between 50% and 55% rate after 60% of the movements.

Table 4.13: The results of the Neural Net DL model compared to the Random Forest ML model for the "all-in" dataset split and the 5th feature set.

	20%	40%	60%	80%	100%
NN	36.40 (6.12)	40.42 (4.58)	45.19 (5.77)	43.63 (6.18)	47.00 (4.19)
RF	**59.15 (5.52)**	**73.43 (4.05)**	**80.40 (3.96)**	**85.45 (2.10)**	**90.35 (3.78)**

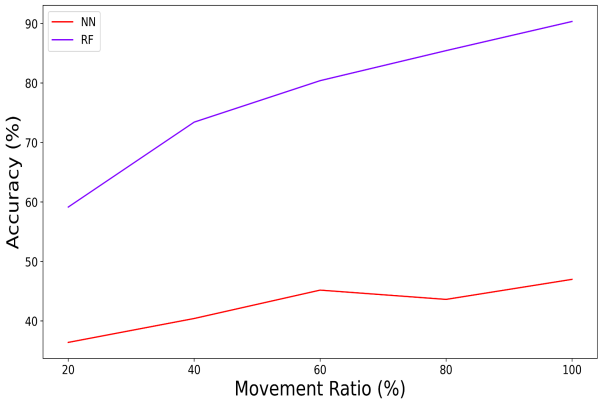

Figure 4.25: The accuracy rate of the Neural Net DL model and of the Random Forest ML model with respect to the movement ratio for the "all-in"dataset split and the 5th feature set.

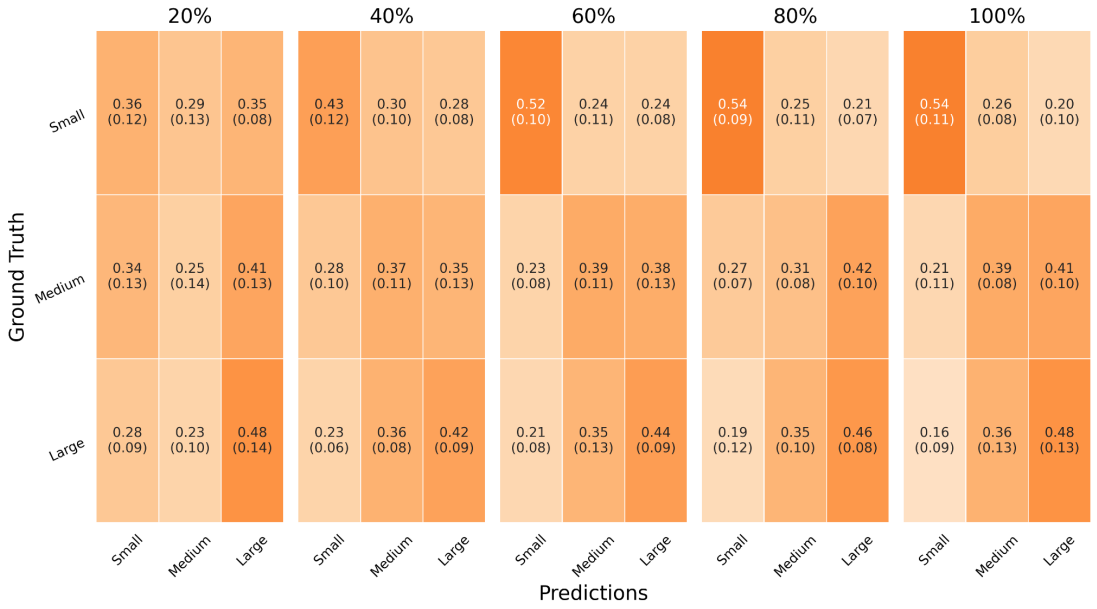

Figure 4.26: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 5th feature set with respect to the movement completion percentages.

The results for the **6th feature set** are presented in Table 4.14 and Fig. 4.27. The feature set consists of the aperture and the wrist speed plane features (Table 3.1). The DL model's accuracy was initially approximately 40% and gradually increased to

approximately 60% for the full movement. Compared to the top-performing Extra Trees ML model, the DL model was consistently outperformed for all the movement completion intervals by a margin of approximately 20% to 30%. Moreover, based on Fig. 4.28, the model classified most of the movements as large regardless of the ground-truth label for the 20% movement completion interval. The medium objects were classified correctly with a better accuracy rate than random guessing only for the 60% and 100% movement completion intervals (35% and 40% respectively). The misclassification of the medium-labeled movements was the main reason for the poor performance of the model. At the same time, the accuracy rate for the small objects increased from 35% to over 70%, while the accuracy rate for the large objects increased from approximately 60% to 65% as the movement progressed.

Table 4.14: The results of the Neural Net DL model compared to the Extra Trees ML model for the "all-in" dataset split and the 6th feature set.

	20%	40%	60%	80%	100%
NN	39.15 (5.10)	41.69 (6.64)	50.64 (5.71)	51.18 (4.25)	59.15 (4.80)
ET	**59.01 (4.36)**	**72.87 (5.46)**	**79.85 (3.16)**	**86.56 (3.87)**	**89.08 (4.17)**

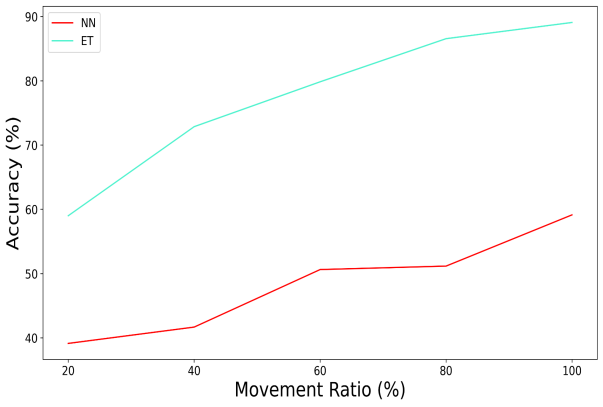

Figure 4.27: The accuracy rate of the Neural Net DL model and of the Extra Trees ML model with respect to the movement ratio for the "all-in"dataset split and the 6th feature set.

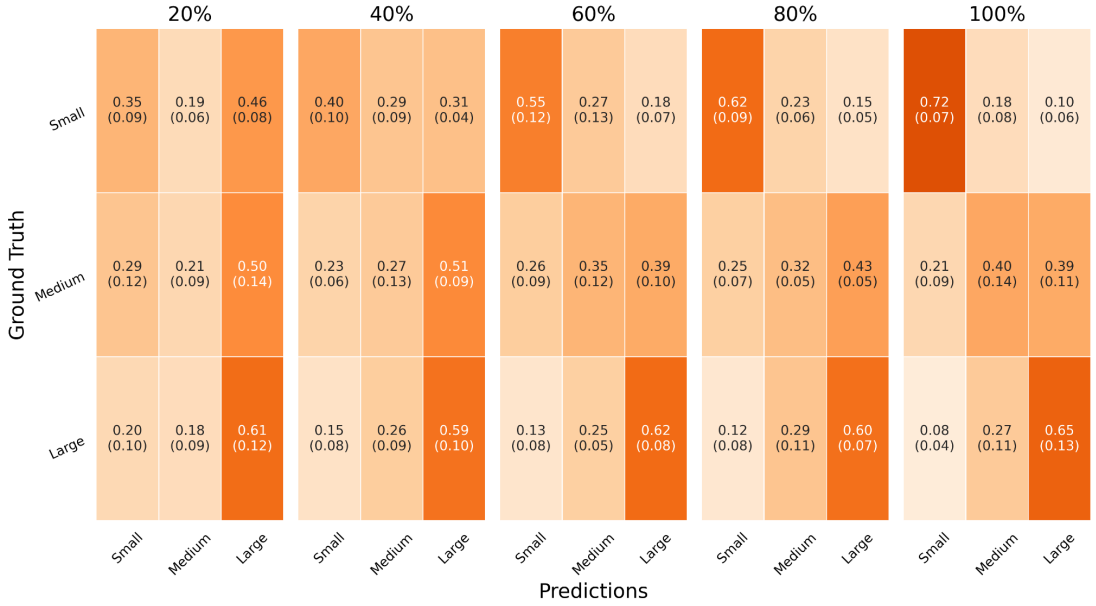

Figure 4.28: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 6th feature set with respect to the movement completion percentages.

The results of the DL model for the **7th feature set** are presented in Table 4.15 and Fig.

4.29. The 7th feature set includes the aperture, the wrist coordinate and the wrist coordinates' dispersion axes features (Table 3.1). The DL model's performance ranged from approximately 45% to 70% for 20% and 100% of the movement respectively. As a result, the model's accuracy rates were at least 20% less than the corresponding accuracy rates of the top-performing Extra Trees ML model. Based on Fig. 4.30, the DL model learned to classify correctly the small-labeled movements as the movement ratio increased, since the accuracy rate increased from approximately 55% to 90%. Moreover, the main reason for the suboptimal performance of the model after 60% was that the model often misclassified the medium object as large and vice-versa. More specifically, at least 30% of the medium objects were misclassified as large and at least 20% of the large objects were misclassified as medium after 60%. On the contrary, for 20% and 40% of the movement, the poor performance of the model was the medium object misclassifications, since the model performed worse than random guessing.

Table 4.15: The results of the Neural Net DL model compared to the Extra Trees ML model for the "all-in" dataset split and the 7th feature set.

	20%	40%	60%	80%	100%
NN	44.19 (5.57)	50.07 (7.16)	61.52 (5.33)	66.44 (5.07)	69.09 (3.12)
ET	**64.90 (3.61)**	**80.41 (3.49)**	**86.28 (2.79)**	**89.37 (3.08)**	**92.86 (2.85)**

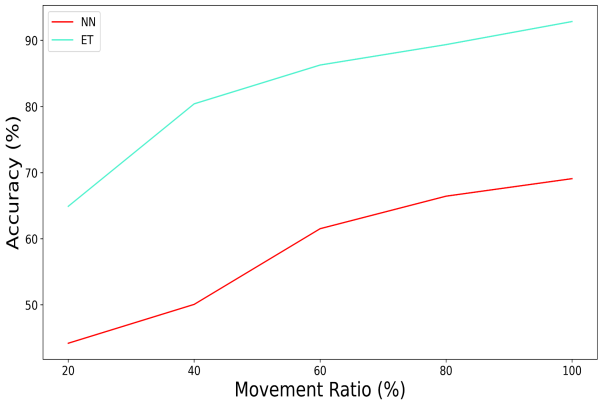

Figure 4.29: The accuracy rate of the Neural Net DL model and of the Extra Trees ML model with respect to the movement ratio for the "all-in"dataset split and the 7th feature set.

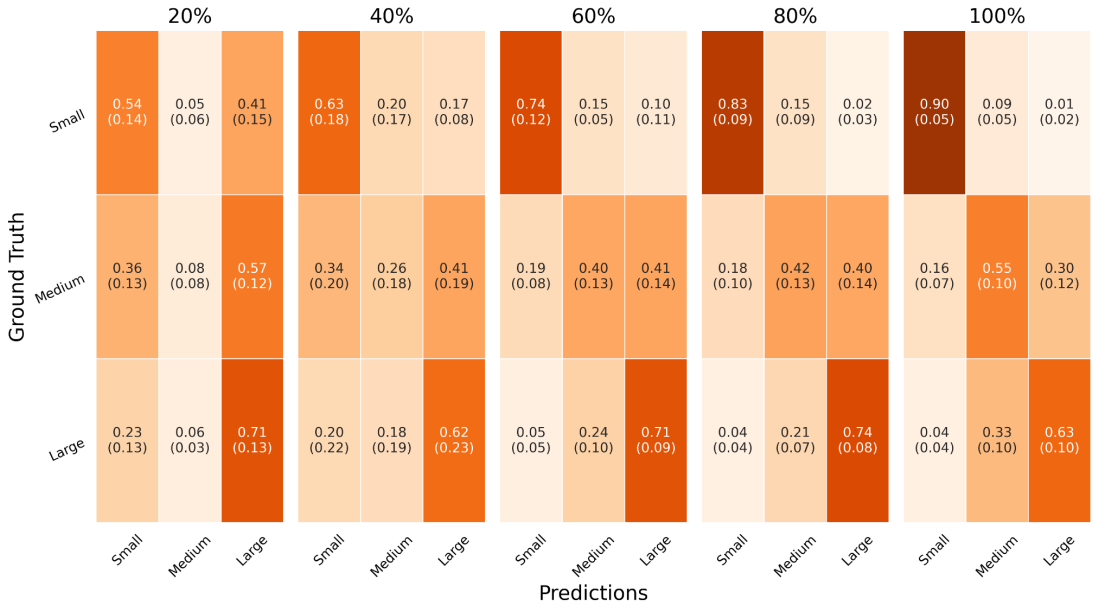

Figure 4.30: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 7th feature set with respect to the movement completion percentages.

The results of the DL model for the **8th feature set** are presented in Table 4.16 and Fig. 4.31. This feature set contains the aperture, the wrist coordinate and the wrist speed axes features (Table 3.1). The model's performance was poor, since the accuracy rates were between 40% and 50% for all the movement completion intervals. Compared to the top-performing Extra Trees ML model, the DL model was outperformed by over 20% and over 40% for 20% and after 40% respectively. Based on Fig 4.32, the model's poor performance can be further explained, since the accuracy rates of all the objects for all the movement completion percentages were less than 55%, with the exception of the large object for the 20% movement completion interval. The reason why this accuracy rate was above 60% was that the majority of the movements were classified as large regardless of their ground-truth label for 20%.

Table 4.16: The results of the Neural Net DL model compared to the Extra Trees ML model for the "all-in" dataset split and the 8th feature set.

	20%	40%	60%	80%	100%
NN	40.16 (5.24)	37.34 (6.08)	43.92 (3.83)	44.78 (6.21)	47.27 (5.62)
ET	**63.64 (4.90)**	**79.43 (3.41)**	**85.99 (5.14)**	**88.81 (2.15)**	**93.00 (1.78)**

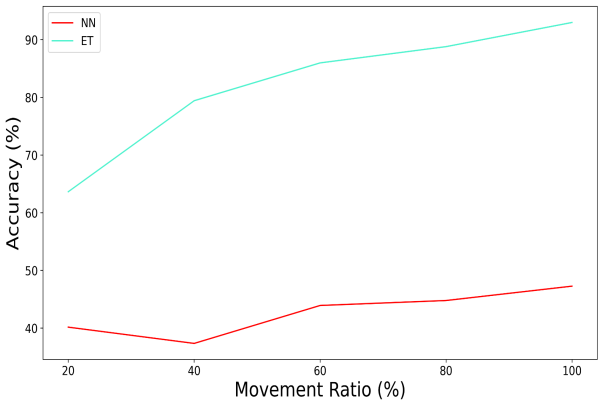

Figure 4.31: The accuracy rate of the Neural Net DL model and of the Extra Trees ML model with respect to the movement ratio for the "all-in"dataset split and the 8th feature set.

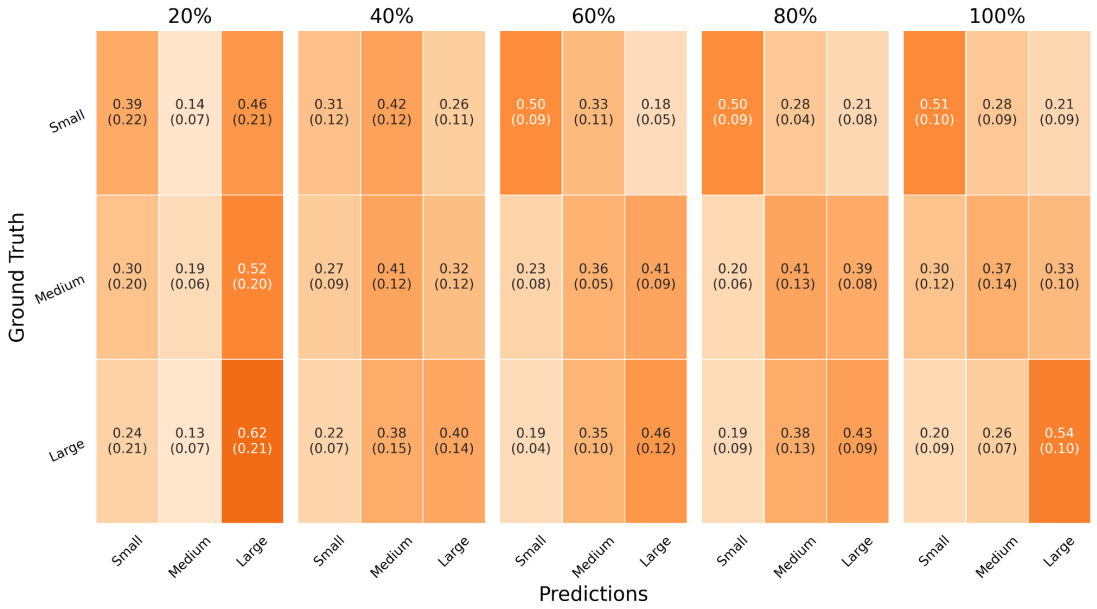

Figure 4.32: The confusion matrices of the Neural Net DL model for the "all-in" dataset split and the 8th feature set with respect to the movement completion percentages.

Conclusively, *the DL model was consistently outperformed by the top-performing ML*

model for all the movement completion intervals and all the feature sets. The model's accuracy rates were higher for the **3rd feature set**. Specifically, the accuracy rates for this feature set were 51.21%, 63.08%, 75.25%, 81.09% and 86.99% for the 20%, 40%, 60%, 80% and 100% movement completion intervals respectively. The 3rd feature set includes the aperture and the wrist coordinates' dispersion axes features. Therefore, due to the latter axes features, the feature set is considered as workspace-dependent. Apart from the 3rd feature set, the model's accuracy was satisfactory for the 1st and the 4th feature sets with the accuracy rates ranging from approximately 50% to below 85% for 20% and 100% respectively. For the 4th feature set, whose results were slightly better, the accuracy rates of the DL model were 49.37%, 63.91%, 74.41%, 80.69% and 83.36% for 20%, 40%, 60%, 80% and 100% respectively. The **1st feature set** includes the aperture features, while the **4th feature set** includes the aperture and the wrist coordinates' dispersion plane features. Therefore, both of these feature sets can be considered as workspace-independent. Additionally, the standard deviations of the DL model for the feature sets 1,3 and 4 were, in general, slightly larger than the corresponding standard deviations of the top-performing ML models.

Furthermore, similar conclusions can be extracted when examining the CMs of the Neural Net DL model for the feature sets 1,3 and 4. *For these feature sets, the DL model learned to classify correctly the small-labeled movements and to avoid the misclassification of the medium- and large-labeled movements as small for the later stages of the movement. As a result, the main reason for the suboptimal performance of the DL model, as the movement progressed, was the misclassification of the medium-labeled movements as large and vice-versa.* Specifically, after 60% the DL model's accuracy rates for the small-labeled movements were approximately 90% or more. On the other hand, the top-performing ML models achieved similar results after 40% of the movement instead of 60%. Moreover, these ML models could identify the large object as non-small even for 20% with an error rate of approximately 15% or less, while the DL model had an error rate of approximately 25%. Finally, the DL model performed worse than random guessing for the medium-labeled movements at the 20% movement completion interval, while the top-performing ML models had an accuracy rate of at least 42% for this task.

4.1.2. "One-out" Results

The results of the "one-out" dataset split strategy are presented following the same structure as in the 4.1.1 subsection. Specifically, the results of the "one-out" strategy for the ML models and the DL model are presented in the subsections 4.1.2.1 and 4.1.2.2 respectively. At the end of each subsection, the results are summarized and commented on.

4.1.2.1. "Traditional" Machine Learning Results

The results of the ML models for the **1st feature set** are presented in Table 4.17 and Fig. 4.33. The 1st feature set includes only the fingertips aperture features (Table 3.1). The top-performing ML model for all the movement completion intervals was **Support Vector Machine**. The model's accuracy was below 50%, below 60%, above 65%, below 65% and below 70% for 20%, 40%, 60%, 80% and 100% of the movement. Therefore, the accuracy rate did not gradually increase as the movement progressed, since the accuracy rate for 80% is less than the accuracy rate for 60%. The standard deviations of the model were approximately 9% or more for all the movement ratios. Based on Fig. 4.34, the Support Vector Machine model learned to identify the small

object as non-large and the large object as non-small after 40% of the movement, since the corresponding error rates were 2% or less and 9% or less. On the other hand, the suboptimal performance of the model after 40% was the result of misclassifications of the medium-labeled movements as either small or large and the misclassifications of the small- and large-labeled movements as medium.

Table 4.17: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 1st feature set.

	20%	40%	60%	80%	100%
RF	45.84 (5.98)	56.50 (6.69)	60.29 (10.66)	64.52 (15.01)	66.62 (19.29)
GB	44.73 (6.21)	56.50 (7.47)	58.45 (10.45)	60.58 (13.49)	64.52 (16.91)
ET	43.03 (8.68)	55.11 (4.67)	61.14 (8.56)	64.81 (12.99)	66.90 (16.05)
SVM	**48.04 (11.38)**	**59.43 (8.95)**	**66.16 (9.16)**	**64.94 (13.42)**	**68.25 (13.76)**
GP	38.75 (8.50)	48.69 (8.00)	57.21 (8.06)	57.93 (9.90)	57.07 (12.20)

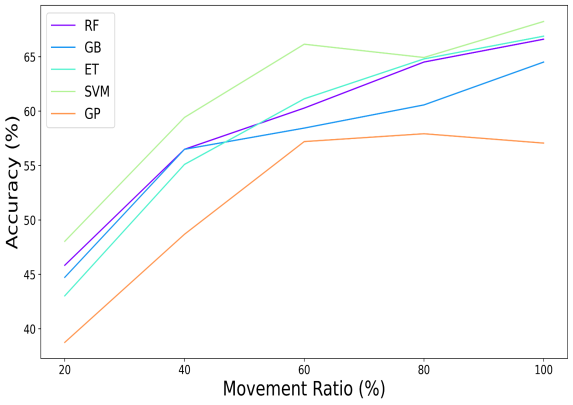

Figure 4.33: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 1st feature set.

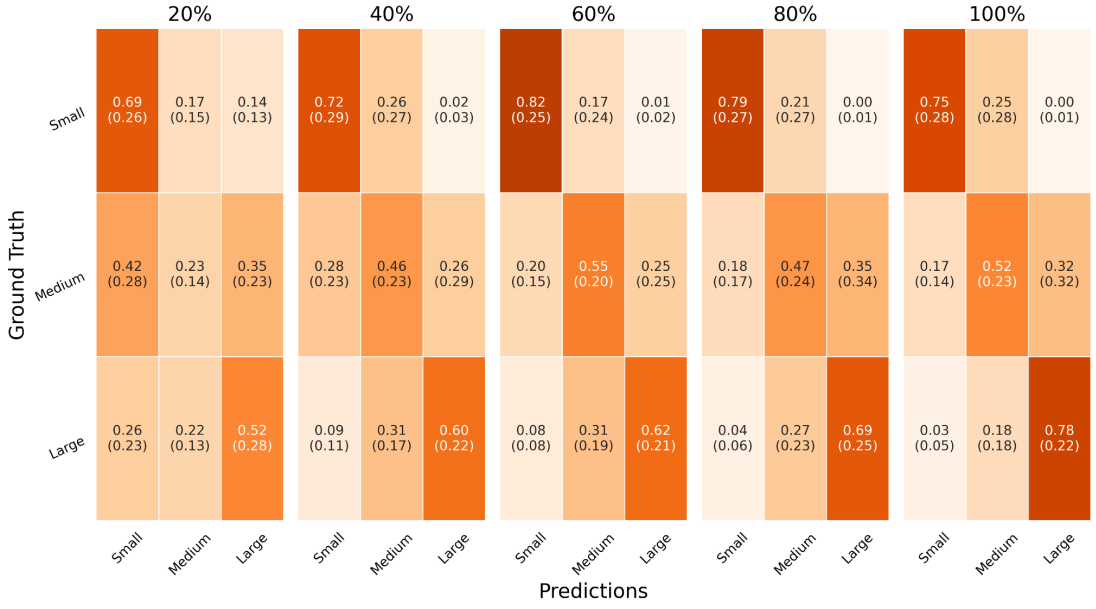

Figure 4.34: The confusion matrices of the Support Vector Machine ML model for the "one-out" dataset split and the 1st feature set with respect to the movement completion percentages.

For the **2nd feature set** the ML models' results are presented in Table 4.18 and Fig. 4.35. The 2nd feature set includes the aperture and the wrist coordinates features (Table 3.1). The top-performing ML models were Random Forest and **Support Vector Machine**. The latter was chosen for further analysis. The accuracy rates of the Support Vector Machine ML model were below 45%, below 55%, over 55%, below 65% and below 70% for 20%, 40%, 60%, 80% and 100% respectively. The model's standard deviations were approximately 10% and at least 13% for 20% and after 40%

respectively. Based on Fig. 4.36, the model's accuracy rate for the small-labeled movements was approximately 80% for all the movement ratios. The model learned to identify the small object as non-large and the large object as non-small with error rates below 1% and 7% after 80%. The medium-labeled movements' accuracy gradually increased from below 25% for 20% to over 70% for the full movement. The main reason for the suboptimal performance of the ML model after 60% was the misclassification of the large objects as medium, with an error rate larger than 40%.

Table 4.18: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 2nd feature set.

	20%	40%	60%	80%	100%
RF	40.11 (7.64)	54.10 (9.56)	60.13 (13.14)	64.09 (17.39)	62.83 (20.02)
GB	39.14 (6.48)	50.91 (8.70)	57.06 (11.36)	56.96 (12.74)	60.19 (19.34)
ET	38.71 (5.35)	53.68 (9.65)	57.76 (13.77)	61.01 (15.17)	63.65 (18.87)
SVM	**43.86 (10.73)**	**53.40 (13.59)**	**57.19 (13.37)**	**63.07 (15.70)**	**68.82 (13.43)**
GP	34.69 (2.71)	35.68 (4.15)	34.97 (2.92)	33.57 (0.53)	33.57 (0.53)

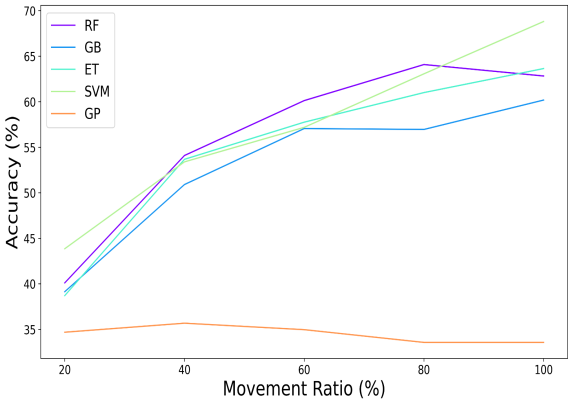

Figure 4.35: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 2nd feature set.

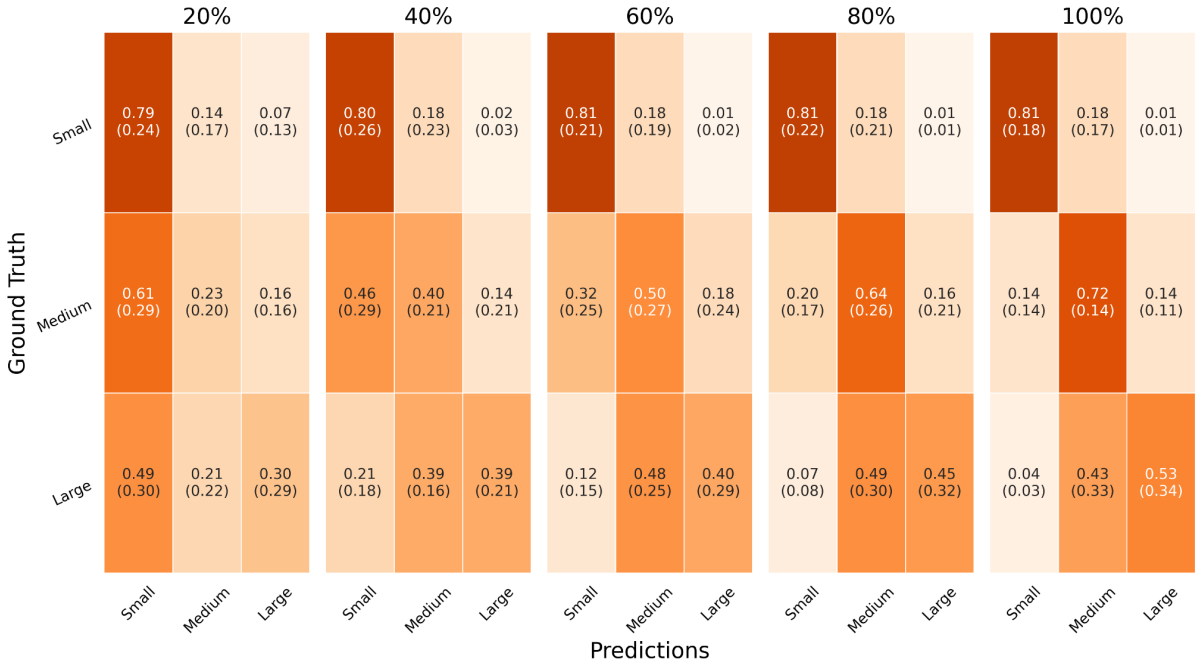

Figure 4.36: The confusion matrices of the Support Vector Machine ML model for the "one-out" dataset split and the 2nd feature set with respect to the movement completion percentages.

For the **3rd feature set** the results of the ML models are presented in Table 4.19 and Fig. 4.37. This feature set consists of the aperture and the wrist coordinates' dispersion axes features (Table 3.1). The top-performing ML models were Random Forest and **Support Vector Machine**. The latter was selected for further analysis. The accuracy

rates of the Support Vector Machine ML model were above 50%, above 60%, above 65%, below 65% and below 70% for 20%, 40%, 60%, 80% and 100% respectively. Therefore, the accuracy did not increase gradually for all the movement completion intervals, since it was less for 80% than for 60%. The model's standard deviations were larger than 9% for all the movement ratios. Based on Fig. 4.38, the model's accuracy rate for the small object was between below 75% and above 80%, the accuracy rate for the medium object was less than 60% and for the large object it ranged between 50% and 80%. The model learned to avoid misclassifying small-labeled movements as large and vice-versa after 40%, with error rates below 2% and 8% respectively.

Table 4.19: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 3rd feature set.

	20%	40%	60%	80%	100%
RF	41.36 (7.81)	57.61 (8.59)	62.54 (11.36)	64.93 (15.41)	67.04 (19.72)
GB	41.23 (7.40)	56.92 (7.72)	56.51 (12.56)	62.84 (14.71)	66.91 (17.94)
ET	41.80 (5.55)	56.93 (7.67)	59.33 (11.00)	64.65 (14.39)	68.71 (17.66)
SVM	**50.83 (14.14)**	**61.67 (9.20)**	**66.58 (10.49)**	**64.93 (12.07)**	**69.08 (13.53)**
GP	38.89 (3.99)	45.48 (10.31)	50.21 (12.30)	53.74 (10.92)	53.58 (12.47)

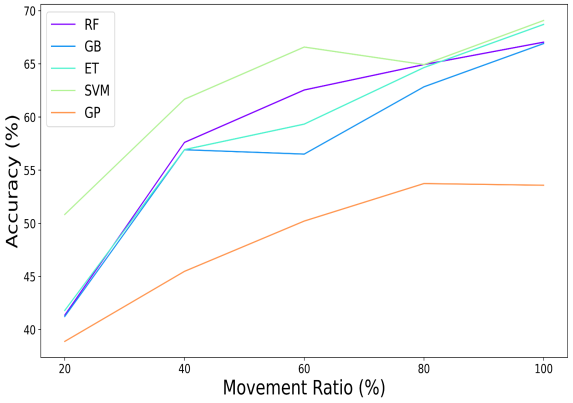

Figure 4.37: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 3rd feature set.

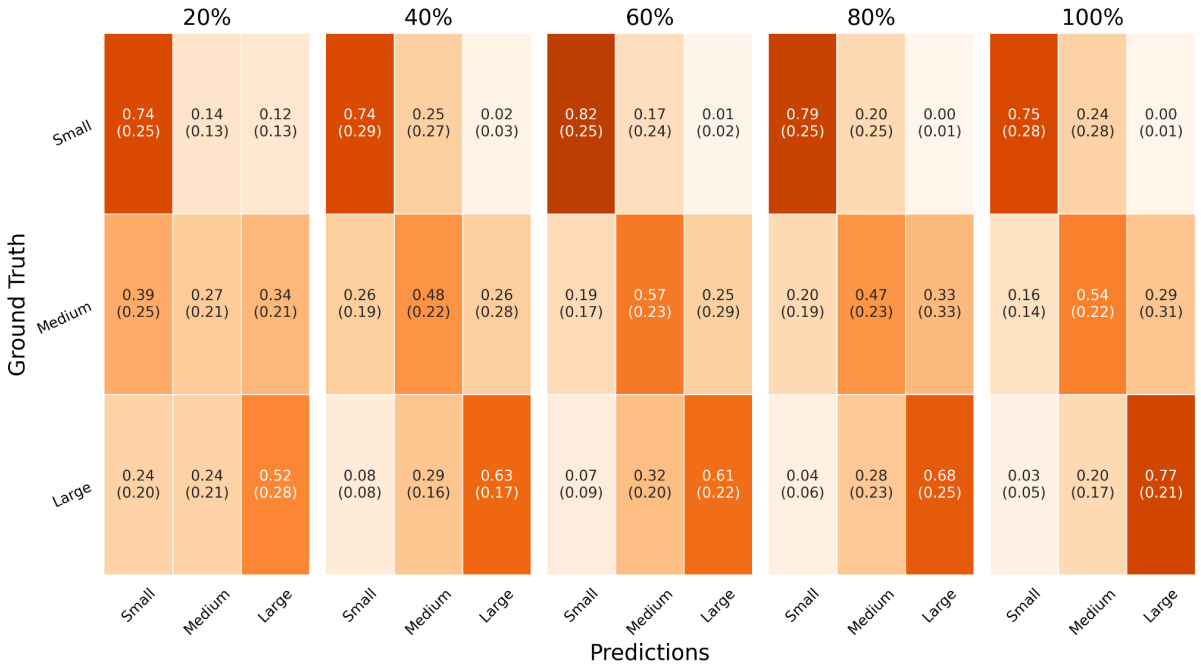

Figure 4.38: The confusion matrices of the Support Vector Machine ML model for the "one-out" dataset split and the 3rd feature set with respect to the movement completion percentages.

The results of the ML models for the **4th feature** set are presented in Table 4.20 and Fig. 4.39. The 4th feature contains the aperture and the wrist coordinates' dispersion

plane features (Table 3.1). The top-performing ML models were Extra Trees and **Support Vector Machine**. The second was selected for further comparisons. The accuracy rates of this model were approximately 50%, 60%, 65%, 65%, 70% for 20%, 40%, 60%, 80% and 100% respectively. The accuracy rate for 80% was slightly less than the accuracy for 60%. The model's standard deviations were approximately between 8.5% and 14%. Based on Fig 4.40, the accuracy rate for the small object was between 74% and 81%, for the medium object it was between 26% and 52% and for the large object between 52% and 78%. After 40% the model learned to identify small objects as non-large and vice-versa with error rates of 2% and 8% at most. For these intervals, the small object and the large object were misclassified as medium with error rates of at least 18% and 19% respectively.

Table 4.20: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 4th feature set.

	20%	40%	60%	80%	100%
RF	44.56 (10.18)	58.74 (8.42)	61.56 (11.52)	63.54 (14.84)	67.03 (18.26)
GB	44.00 (10.52)	54.52 (10.91)	57.35 (11.45)	60.32 (15.98)	64.25 (16.72)
ET	42.63 (7.30)	57.49 (7.90)	61.16 (9.39)	65.23 (15.36)	67.04 (17.60)
SVM	**50.69 (14.08)**	**61.38 (8.63)**	**64.64 (9.62)**	**64.38 (11.53)**	**68.80 (13.90)**
GP	40.69 (5.16)	47.45 (10.15)	53.55 (12.10)	55.83 (10.74)	56.09 (12.77)

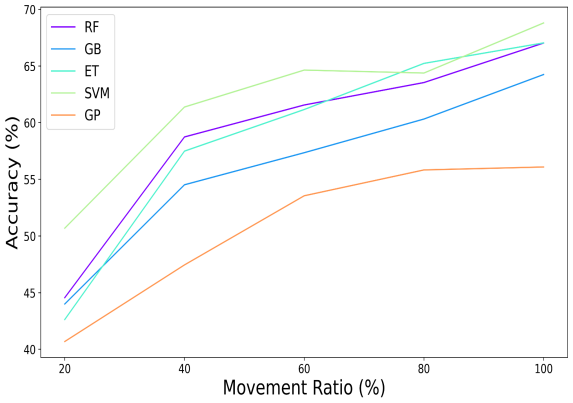

Figure 4.39: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 4th feature set.

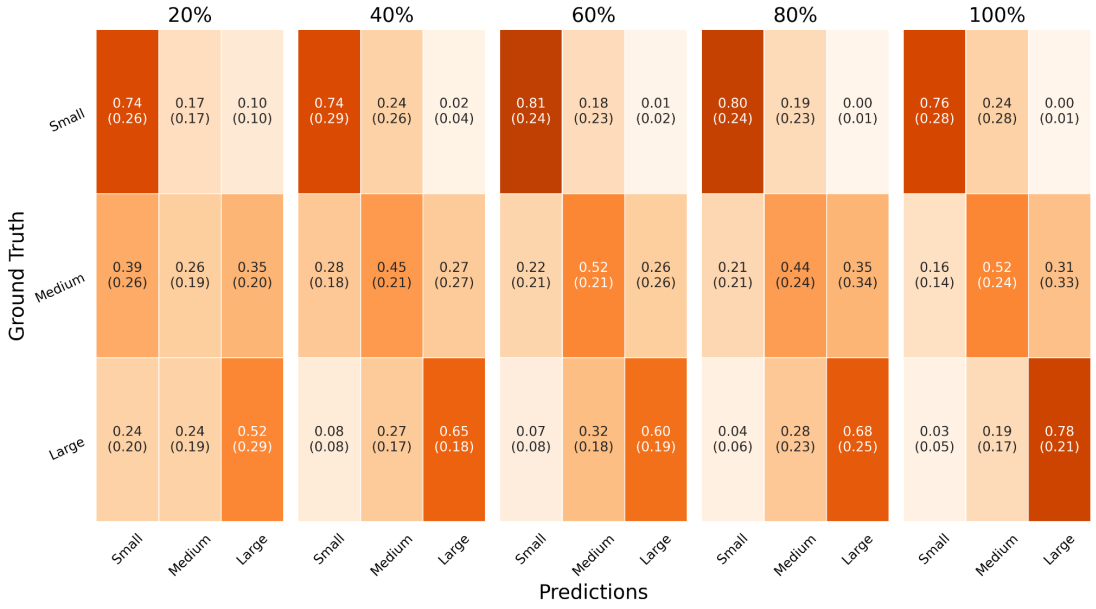

Figure 4.40: The confusion matrices of the Support Vector Machine ML model for the "one-out" dataset split and the 4th feature set with respect to the movement completion percentages.

The results of the ML models for the **5th feature set** are presented in Table 4.21 and Fig. 4.41. The 5th feature set includes the aperture and the wrist speed axes features (Table 3.1). The top-performing ML models for this feature set were **Random Forest** and Extra Trees. The former was chosen to be further analyzed. The accuracy rates of the model gradually increased from approximately 45% for 20% to above 65% for the full movement. Based on Fig. 4.42, the model learned to identify the small object as non-large after 40% with an error rate of 8% at most. Similarly the model learned to identify the large object as non-small after 60% with an error rate of 7% at most. At the same time, the small and the large were misclassified as medium with error rates of at least 17%. The accuracy rate for the medium labeled movements ranged between 24% and 46%, with the majority of the misclassified movements predicted to be large.

Table 4.21: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 5th feature set.

	20%	40%	60%	80%	100%
RF	**45.14 (4.69)**	**56.92 (5.61)**	**60.71 (10.25)**	**65.09 (13.95)**	**66.48 (17.22)**
GB	41.65 (5.62)	54.00 (6.67)	56.08 (10.98)	59.89 (13.83)	66.07 (15.27)
ET	43.19 (6.36)	55.39 (8.04)	59.74 (9.50)	65.64 (13.19)	68.15 (16.47)
SVM	30.65 (4.55)	40.44 (3.98)	48.28 (8.10)	54.55 (7.46)	51.89 (4.30)
GP	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)

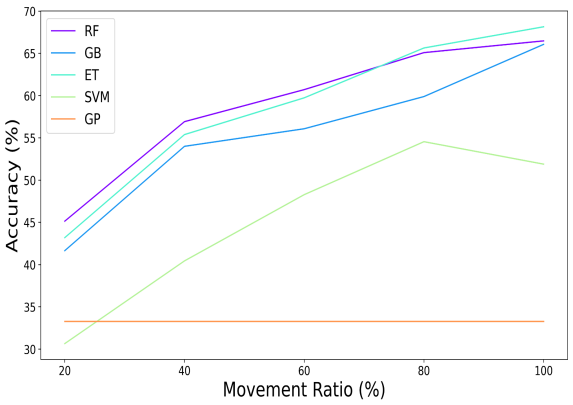

Figure 4.41: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 5th feature set.

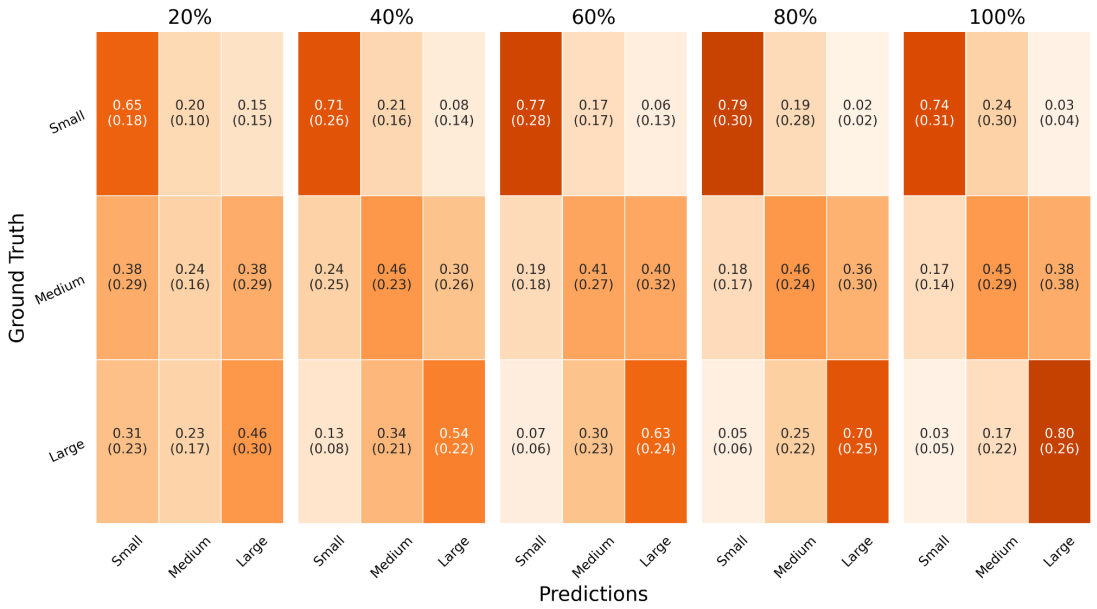

Figure 4.42: The confusion matrices of the Random Forest ML model for the "one-out" dataset split and the 5th feature set with respect to the movement completion percentages.

For the **6th feature set**, the results of the ML models are presented in Table 4.22 and

Fig. 4.43. The feature set consists of the aperture and the wrist speed plane features (Table 3.1). The top-performing ML models were **Random Forest** and Extra Trees. The first one was selected for Fig. 4.44 and the DL comparisons. As the movement progressed, the accuracy rates and the standard deviations gradually increased from above 45% to above 65% and from approximately 7.5% to 18.5% respectively. Based on Fig. 4.44, the accuracy rates for the small and the large object ranged between 69% and 79% and between 46% and 79% respectively. The model misclassified the small object as large after 40% with a small error rate between 1% and 8%. Similarly, the large object was misclassified as small after 60% with an error rate between 3% and 6%. The accuracy rate for the medium object ranged from 20% to 48% and the most misclassified medium movements were predicted to be large.

Table 4.22: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 6th feature set.

	20%	40%	60%	80%	100%
RF	**45.14 (7.55)**	**57.20 (7.17)**	**61.41 (10.95)**	**62.70 (13.76)**	**67.04 (18.65)**
GB	44.30 (6.05)	54.69 (6.26)	58.18 (10.18)	61.56 (13.28)	64.94 (16.50)
ET	43.75 (6.48)	54.27 (5.92)	60.16 (9.44)	65.94 (13.59)	66.21 (17.41)
SVM	32.32 (5.22)	39.87 (6.70)	51.47 (10.72)	56.63 (9.41)	56.93 (5.68)
GP	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)

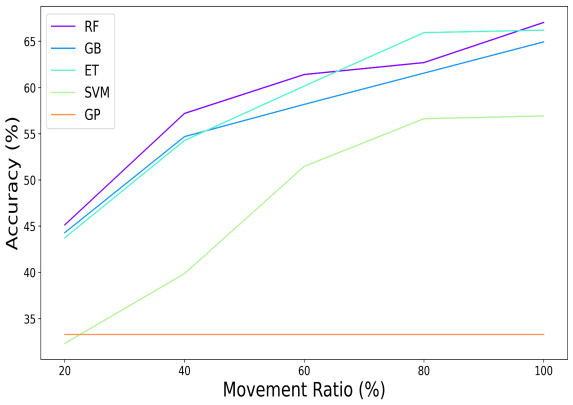

Figure 4.43: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 6th feature set.

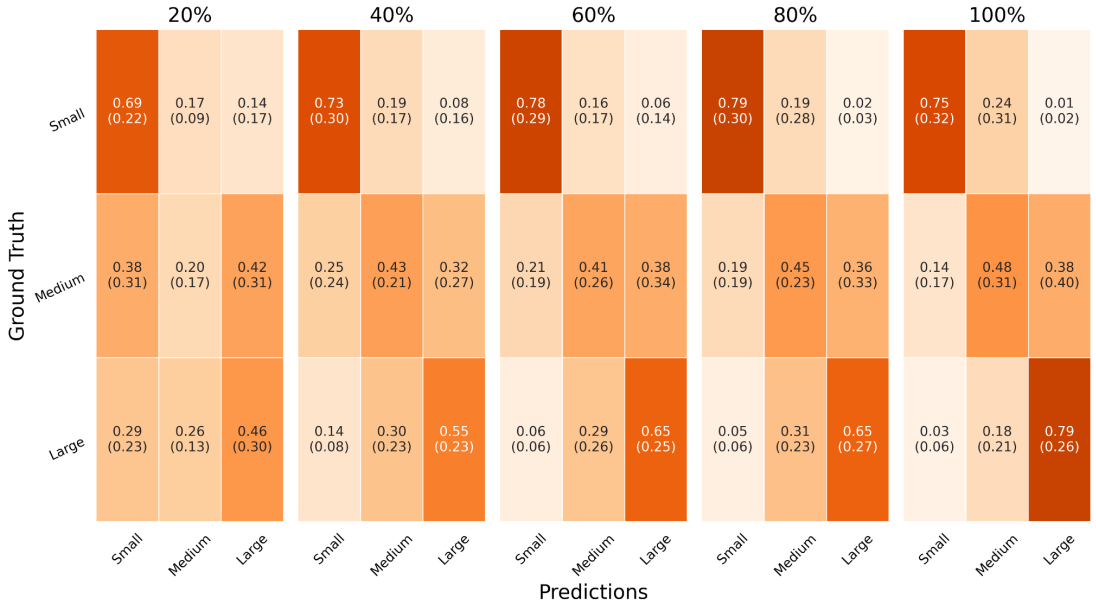

Figure 4.44: The confusion matrices of the Random Forest ML model for the "one-out" dataset split and the 6th feature set with respect to the movement completion percentages.

For the **7th feature set** the results of the ML models are presented in Table 4.23 and

Fig. 4.45. The 7th feature set includes the aperture, the wrist coordinate and the wrist coordinates' dispersion axes features (Table 3.1). The top-performing ML models were **Random Forest** and Support Vector Machine. The former model was chosen for further analysis. This model's accuracy rates ranged from above 40% to below 65%, while the standard deviations were between 8% and 19%. The accuracy did not gradually increase throughout the movement, since the accuracy rate for 80% was higher than the accuracy rate for 100%. Based on Fig. 4.46, the medium movements had accuracy 16% for 20% and around 45% after 40%. The majority of the misclassified medium objects were predicted to be large. The accuracy rate for the small object was 58% for 20% and between 72% and 77% after 40% and the majority of the misclassified small movements were predicted to be medium. The latter can be observed also for the large object, for which the accuracy rates gradually increased from 48% to 74%.

Table 4.23: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 7th feature set.

	20%	40%	60%	80%	100%
RF	**40.80 (9.38)**	**55.78 (8.17)**	**61.84 (11.81)**	**64.24 (18.60)**	**63.95 (18.44)**
GB	40.24 (9.33)	52.44 (10.91)	55.82 (11.66)	56.94 (13.70)	60.61 (17.83)
ET	39.69 (7.19)	53.67 (7.97)	57.62 (10.98)	62.00 (18.47)	64.92 (17.81)
SVM	43.45 (10.44)	52.56 (13.09)	57.19 (13.76)	62.94 (15.03)	69.53 (13.60)
GP	33.85 (1.07)	35.25 (3.34)	34.69 (2.45)	33.57 (0.53)	33.57 (0.53)

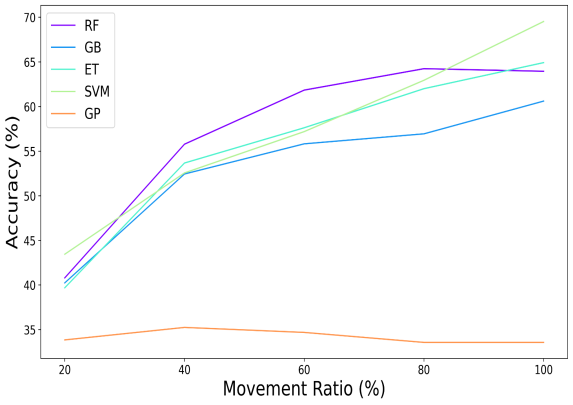

Figure 4.45: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 7th feature set.

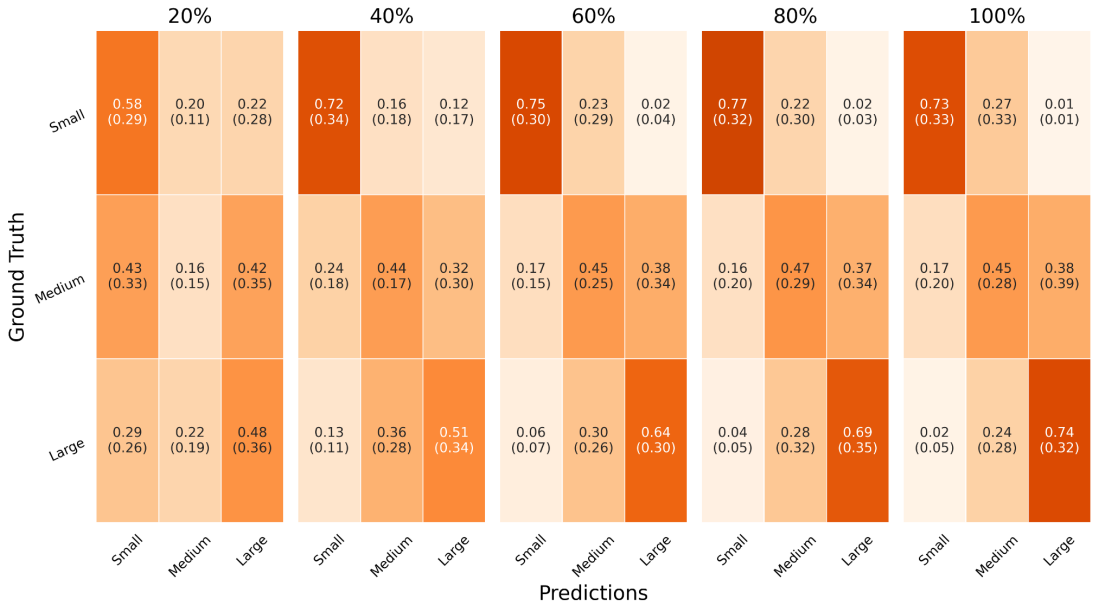

Figure 4.46: The confusion matrices of the Random Forest ML model for the "one-out" dataset split and the 7th feature set with respect to the movement completion percentages.

The ML models' results for the **8th feature set** are presented in Table 4.24 and Fig. 4.47. The 8th feature set includes the aperture, the wrist coordinate and the wrist speed axes features (Table 3.1). The top-performing ML model was **Random Forest** for all the movement completion intervals with the accuracy rate gradually increasing from over 40% for 20% to below 65% for 100%. Simultaneously, the standard deviations also increased gradually from below 7% to 19% as the movement progressed. Based on Fig. 4.48, the accuracy rate for the small- and the large-labeled movements ranged between 62% and 77% and between 50% and 71% respectively. The model learned to identify the small object as non-large and the large object as non-small with a maximum error rate of 6% after 60%. Therefore the suboptimal accuracy for these objects was due to misclassifying them as medium. The accuracy for the medium object was 12% for 20% and between 40% and 50% after 40% of the movement.

Table 4.24: The evaluation of the "traditional" ML models for the "one-out" dataset split and the 8th feature set.

	20%	40%	60%	80%	100%
RF	**41.64 (6.75)**	**56.22 (9.13)**	**59.04 (13.40)**	**63.10 (17.52)**	**64.79 (19.00)**
GB	41.23 (6.40)	52.32 (8.88)	55.81 (11.64)	57.25 (14.88)	58.38 (17.34)
ET	38.43 (7.53)	50.05 (9.32)	58.34 (11.66)	61.15 (18.21)	62.96 (16.91)
SVM	30.65 (4.40)	39.88 (4.04)	45.62 (8.34)	50.50 (4.37)	47.57 (6.66)
GP	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)	33.29 (0.35)

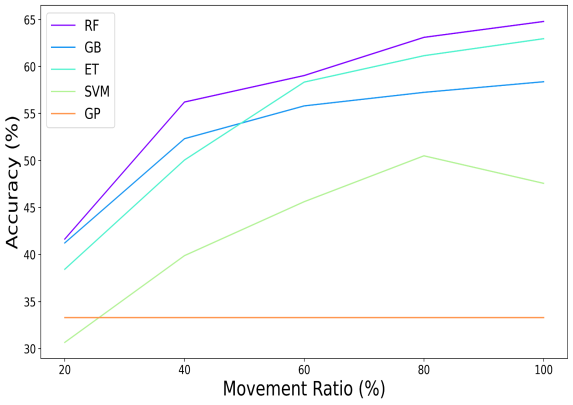

Figure 4.47: The accuracy rates of the "traditional" ML models with respect to the movement ratio for the "one-out" dataset split and the 8th feature set.

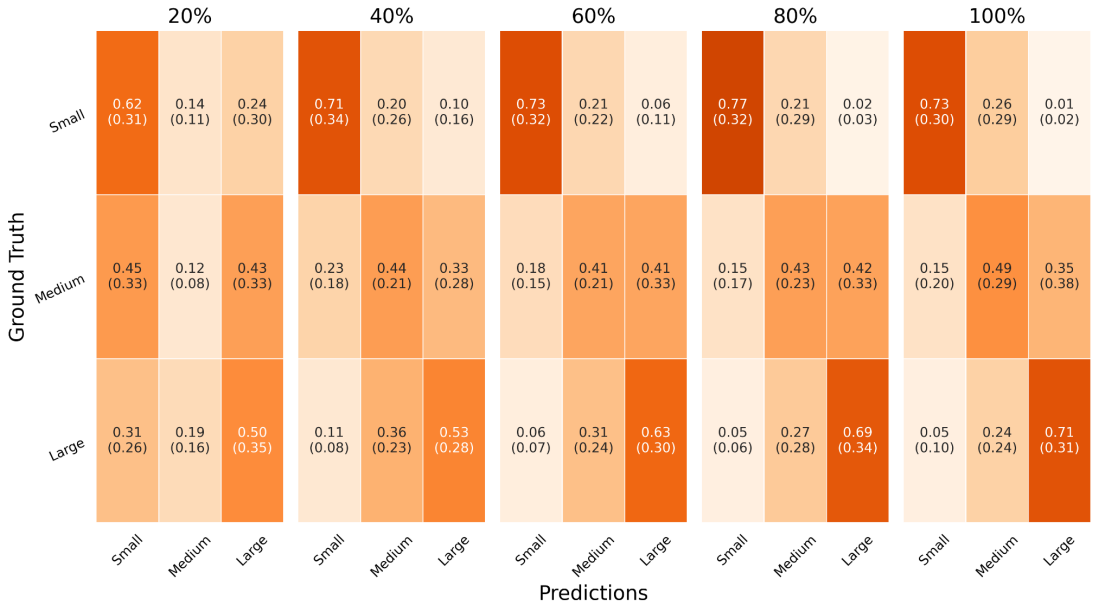

Figure 4.48: The confusion matrices of the Random Forest ML model for the "one-out" dataset split and the 8th feature set with respect to the movement completion percentages.

In summary, the top-performing ML models for the “one-out” dataset split strategy were *Random Forest*, *Extra Trees* and *Support Vector Machine*. The best accuracy rates were logged for the **Support Vector Machine** model and the **3rd feature set**. Specifically, in this case the accuracy rates were 50.83%, 61.67%, 66.58%, 64.93% and 69.08% for 20%, 40%, 60%, 80% and 100% respectively. Since the 3rd feature set contains the aperture and the wrist coordinates’ dispersion axes features, it is considered to be a workspace-dependent feature set. Regarding the workspace-independent feature sets, the best accuracy rates were observed for the Support Vector Machine ML model and for the 1st and the 4th features sets. For the **1st feature set**, which contains the aperture features, the accuracy rates were 48.04%, 59.43%, 66.16%, 64.94% and 68.25% for 20%, 40%, 60%, 80% and 100%. For the **4th feature set**, which contains the aperture and the wrist coordinates’ dispersion plane features, the accuracy rates were 50.69%, 61.38%, 64.64%, 64.38% and 68.80% for 20%, 40%, 60%, 80% and 100%. Moreover, one can observe that for these feature sets the standard deviations of the Support Vector Machine model were less than 15%, while the standard deviations of the other top-performing models were generally higher. Finally, compared to the top-performing models for the “all-in” strategy, the accuracy of the top-performing ML models for the “one-out” strategy was significantly lower, while the standard deviation was higher.

The accuracy of the top-performing ML models for the “one-out” strategy can be further analyzed based on the CMs. The CMs of the Support Vector Machine model for the feature sets 1,3 and 4 reveal that the models learn to identify the small object as non-large and the large object as non-small with a maximum error rate of 2% and 9% respectively after 40% of the movement. Nevertheless, the accuracy rates for the small- and the large-labeled movements were approximately 80% or less for all the movement completion intervals. The reason for the suboptimal accuracy of the models for these objects was that they were often misclassified as medium. Furthermore, the models performed worse than random guessing for the medium object for 20%. After 40%, the accuracy rate for this object ranged between 45% and 57%, since the object was misclassified as either small or large. As the movement progressed, the most misclassified medium-labeled movements were predicted to be large. However, the error rate of the medium-labeled movements that were misclassified as small did not drop below 40%.

4.1.2.2. Deep Learning Results

The results of the DL model for the **1st feature set** are presented in Table 4.25 and Fig. 4.49. The 1st feature set contains solely the aperture features (Table 3.1). The accuracy rates of the ML model were below 40%, over 45%, below 50%, over 55% and below 65% for 20%, 40%, 60%, 80% and 100% respectively. Compared to the top-performing Support Vector Machine ML model, the Neural Net DL model was outperformed for all the movement completion intervals. Specifically, with the exception of the 100% movement ratio, the accuracy rate of the Support Vector Machine model was superior by apporximately 10% or more. Based on Fig. 4.50, the model learned to identify the small object as non-large and the large object as non-small after 80% of the movement with maximum error rates 9% and 3%. Moreover for 100%, the model learned to identify the medium object as non-small with an error rate of 8%. The accuracy rates for the small object ranged between 53% and 73%, for the medium object they ranged between 15% and 53% and for the large object they ranged between 49% and 67%. The suboptimal performance of the model in the later stages was primarily due to

misclassifying the medium object as large and secondarily due to misclassifying the small or the large object as medium.

Table 4.25: The results of the Neural Net DL model compared to the Support Vector Machine ML model for the "one-out" dataset split and the 1st feature set.

	20%	40%	60%	80%	100%
NN	38.85 (7.39)	46.82 (10.84)	48.38 (7.81)	55.83 (10.77)	64.47 (12.71)
SVM	**48.04 (11.38)**	**59.43 (8.95)**	**66.16 (9.16)**	**64.94 (13.42)**	**68.25 (13.76)**

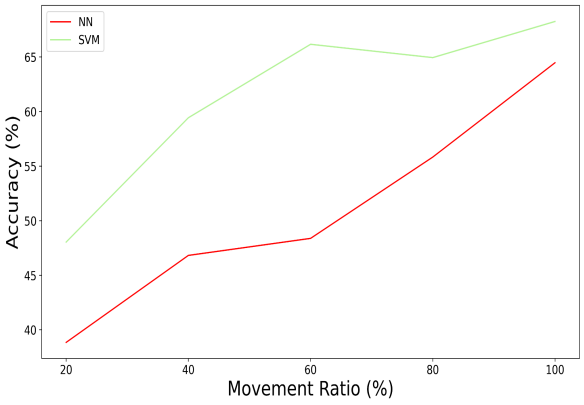

Figure 4.49: The accuracy rates of the Neural Net DL model and of the Support Vector Machine ML model with respect to the movement ratio for the "one-out" dataset split and the 1st feature set.

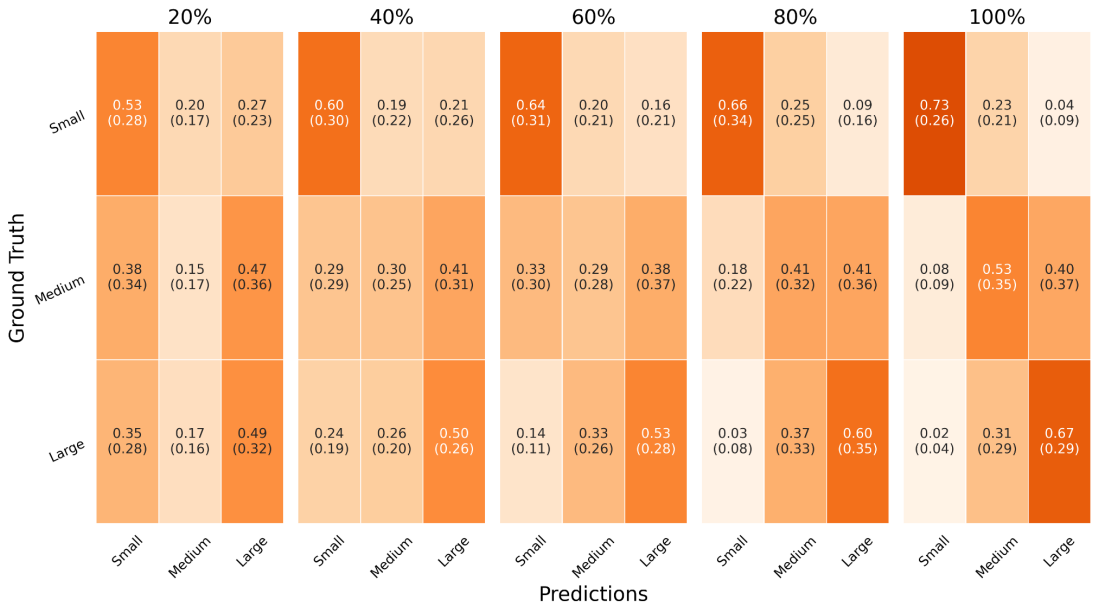

Figure 4.50: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 1st feature set with respect to the movement completion percentages.

For the **2nd feature** set the DL model's results are presented in Table 4.26 and Fig. 4.51. This feature set consists of the aperture and the wrist coordinates features (Table 3.1). The model's accuracy rates ranged from above 35% to below 45% throughout the movement. Compared to the top-performing Support Vector Machine ML model, the DL model was outperformed by a margin of approximately 10% for 20%, 15% for 40% and 60% and 25% for 80% and 100%. The poor performance of the DL model is also evident in the CMs of Fig. 4.52. The accuracy rate for the small-, the medium- and the large-labeled movements was at most 60%, 40% and 55% for all movement completion percentages. Additionally, the model performed worse than random guessing for the medium object for all the movement completion intervals with the exception of 100%.

Table 4.26: The results of the Neural Net DL model compared to the Support Vector Machine ML model for the "one-out" dataset split and the 2nd feature set.

	20%	40%	60%	80%	100%
NN	35.94 (6.93)	35.23 (8.11)	43.33 (15.29)	38.90 (9.32)	42.78 (11.73)
SVM	**43.86 (10.73)**	**53.40 (13.59)**	**57.19 (13.37)**	**63.07 (15.70)**	**68.82 (13.43)**

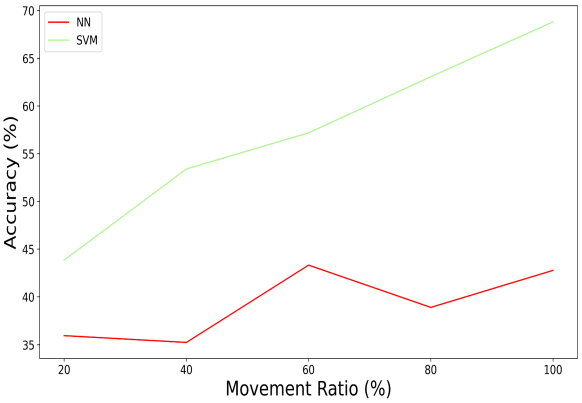

Figure 4.51: The accuracy rates of the Neural Net DL model and of the Support Vector Machine ML model with respect to the movement ratio for the "one-out" dataset split and the 2nd feature set.

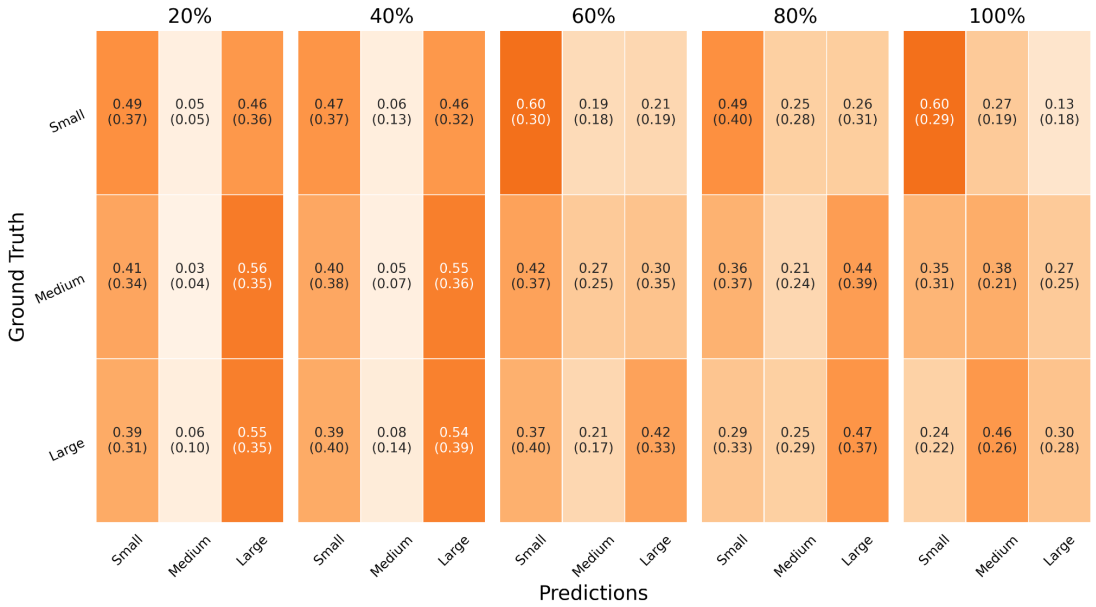

Figure 4.52: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 2nd feature set with respect to the movement completion percentages.

For the **3rd feature set** the results of the Neural Net DL model are presented in Table 4.27 and Fig. 4.53. The feature set includes the aperture and the wrist coordinates' dispersion axes features (Table 3.1). The accuracy rates of the model gradually increased from approximately 40% for 20% to approximately 60% for the full movement. At the same time, the accuracy rates of the top-performing Support Vector Machine model were superior to the accuracy rates of the DL model by approximately 10% for each movement ratio. Based on the CMs of Fig 4.54, one can observe that the DL model learned to identify the large object as non-small after 60% with a maximum error rate of 8%. Similarly, the small object was seldomly classified as large after 80% with an error rate of 7%. Finally, for the full movement, only 10% of the medium-labeled movements were classified as small. Nevertheless, the accuracy rates for the small, the medium and the large object were less than or equal to 71%, 54% and 65% respectively for all the movement completion intervals. The primary reason for these suboptimal accuracies at the later stages of the movement were the medium movements which

were misclassified as large, the large movements which were misclassified as medium and the small movements which were misclassified as medium.

Table 4.27: The results of the Neural Net DL model compared to the Support Vector Machine ML model for the "one-out" dataset split and the 3rd feature set.

	20%	40%	60%	80%	100%
NN	39.99 (8.43)	45.98 (9.58)	53.58 (9.81)	56.94 (7.85)	60.16 (14.34)
SVM	**50.83 (14.14)**	**61.67 (9.20)**	**66.58 (10.49)**	**64.93 (12.07)**	**69.08 (13.53)**

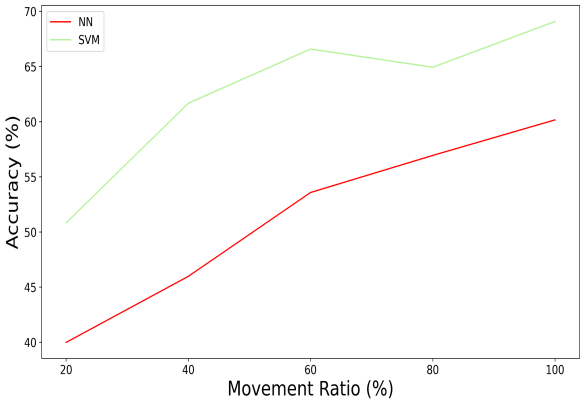

Figure 4.53: The accuracy rates of the Neural Net DL model and of the Support Vector Machine ML model with respect to the movement ratio for the "one-out" dataset split and the 3rd feature set.

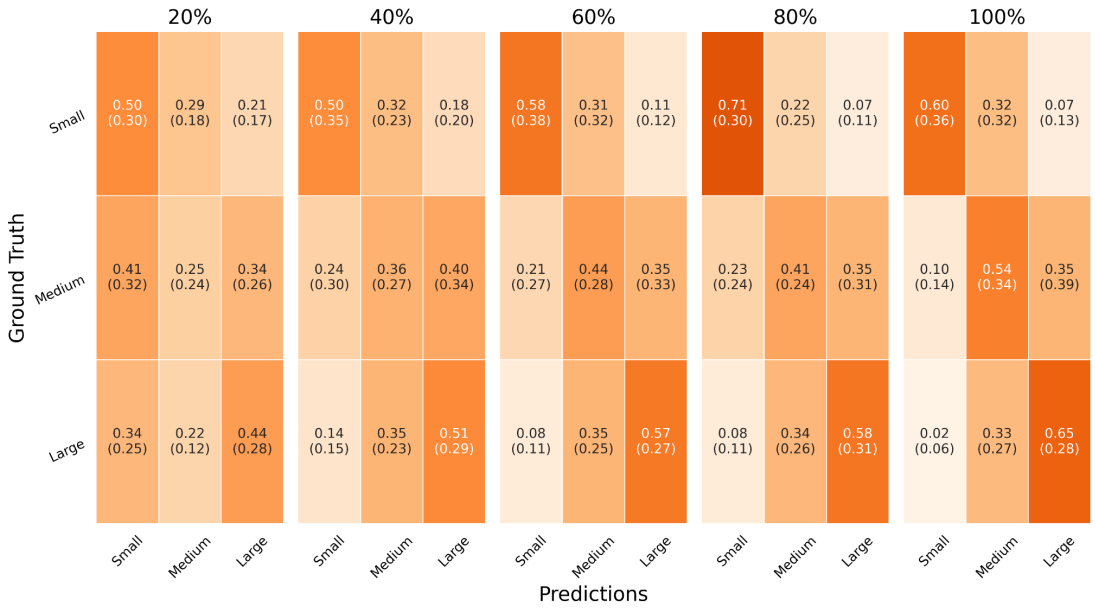

Figure 4.54: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 3rd feature set with respect to the movement completion percentages.

The results of the DL model for the **4th feature set** are presented in Table 4.28 and Fig. 4.55. The 4th feature set includes the aperture and the wrist coordinates' dispersion plane features (Table 3.1). The DL model's accuracy rates ranged from below 40% for 20% to approximately 65% for 100% of the movement. Compared to the top-performing Support Vector Machine ML model, the accuracy rates of the DL model were lower for all the movement completion intervals. Based on Fig. 4.56, the DL model learned to identify the small object as non-large after 80% of the movement with an error rate of 7% or less. Similarly, the model learned to identify the large object as non-small after 80% with an error rate of 2% or less. The accuracy rates for the small object gradually increased from 50% to 76%, for the medium object they ranged between 16% and 48% and, finally, for the large object they gradually increased from 47% to 75%. After 60% at

least 24% of the large movements and 21% of the small movements were misclassified as medium. Moreover, the majority of the misclassified medium movements were predicted to be large for all the movement completion intervals with a minimum error rate of 34%.

Table 4.28: The results of the Neural Net DL model compared to the Support Vector Machine ML model for the "one-out" dataset split and the 4th feature set.

	20%	40%	60%	80%	100%
NN	37.75 (7.84)	50.46 (7.89)	50.21 (10.04)	61.27 (7.66)	65.03 (10.75)
SVM	**50.69 (14.08)**	**61.38 (8.63)**	**64.64 (9.62)**	**64.38 (11.53)**	**68.80 (13.90)**

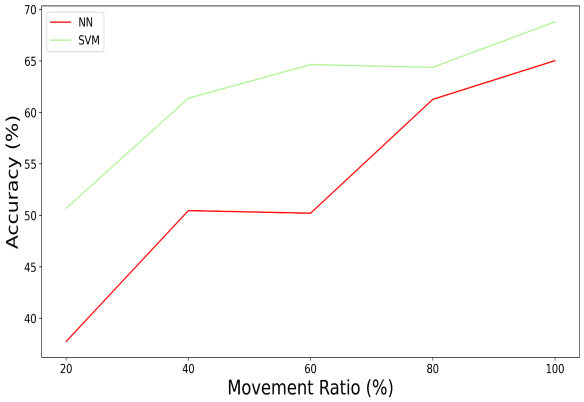

Figure 4.55: The accuracy rates of the Neural Net DL model and of the Support Vector Machine ML model with respect to the movement ratio for the "one-out" dataset split and the 4th feature set.

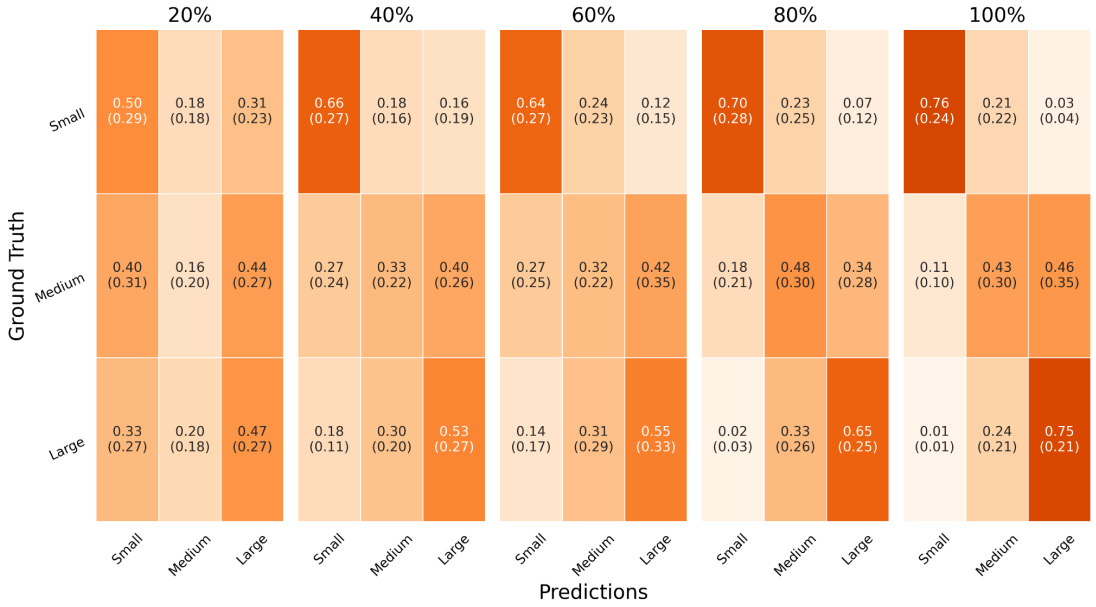

Figure 4.56: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 4th feature set with respect to the movement completion percentages.

The results of the DL model for the **5th feature set** are presented in Table 4.29 and Fig. 4.57. This feature set contains the aperture and the wrist speed axes features (Table 3.1). The accuracy rates of the DL model were between 30% and 40% for all the movement completion intervals. Compared to the top-performing Random Forest ML model, the accuracy rates of the DL model were lower by a margin of at least 15%. The poor performance of the DL model can be further analyzed in the CMs of Fig. 4.58. The accuracy rates of the DL model for the small, the medium and the large object were lower than 50%, 30% and 45% respectively for all the movement completion percentages.

Table 4.29: The results of the Neural Net DL model compared to the Random Forest ML model for the "one-out" dataset split and the 5th feature set.

	20%	40%	60%	80%	100%
NN	30.05 (4.61)	31.46 (4.70)	38.05 (6.37)	34.38 (10.03)	38.76 (8.34)
RF	**45.14 (4.69)**	**56.92 (5.61)**	**60.71 (10.25)**	**65.09 (13.95)**	**66.48 (17.22)**

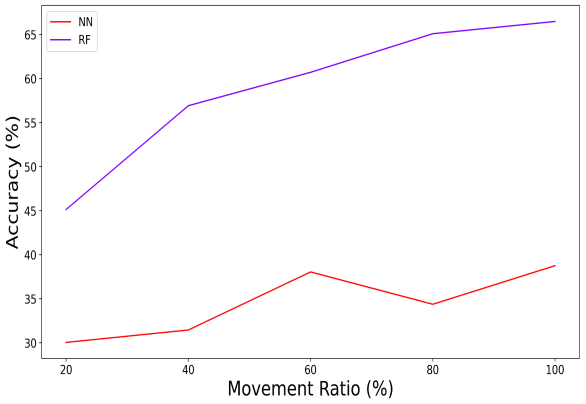

Figure 4.57: The accuracy rates of the Neural Net DL model and of the Random Forest ML model with respect to the movement ratio for the "one-out" dataset split and the 5th feature set.

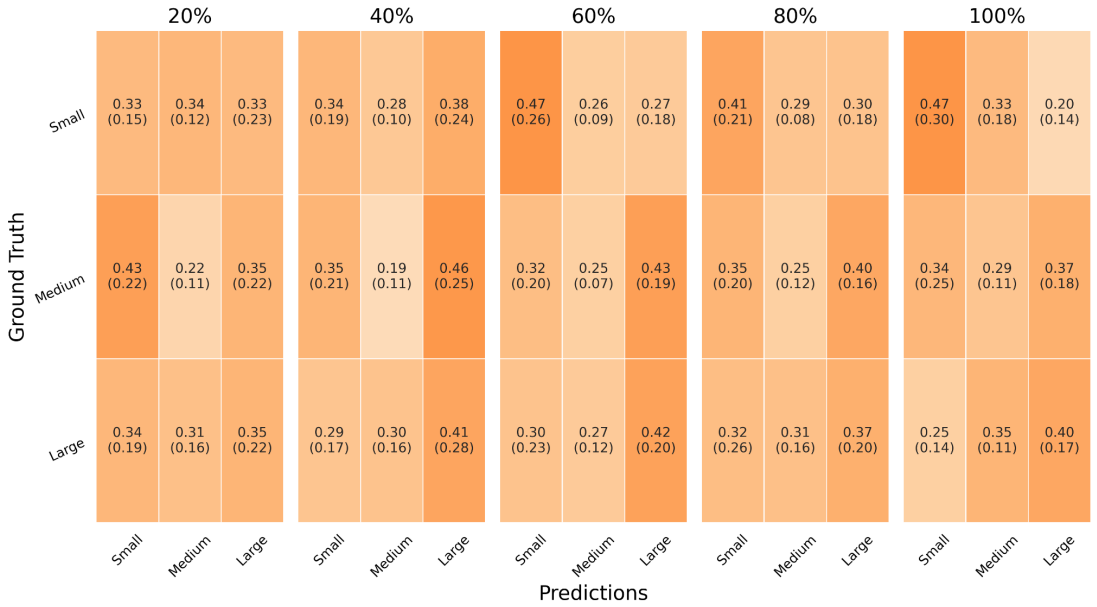

Figure 4.58: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 5th feature set with respect to the movement completion percentages.

For the **6th feature set** the accuracy rates of the DL model are presented in Table 4.30 and Fig. 4.59. The 6th feature set includes the aperture and the wrist speed plane features (Table 3.1). The accuracy rates of the model were between 30% and 40% for all the movement completion percentages. These accuracy rates were significantly worse than the accuracy rates of the top-performing Random Forest ML model. The poor performance of the model can be further analyzed based on the CMs of Fig. 4.60. Specifically, the accuracy rates for the small, the medium and the large object were at most 52%, 29% and 58% respectively for all the movement completion intervals.

Table 4.30: The results of the Neural Net DL model compared to the Random Forest ML model for the "one-out" dataset split and the 6th feature set.

	20%	40%	60%	80%	100%
NN	33.03 (6.68)	39.32 (10.78)	37.88 (11.55)	40.44 (11.65)	43.37 (11.14)
RF	**45.14 (7.55)**	**57.20 (7.17)**	**61.41 (10.95)**	**62.70 (13.76)**	**67.04 (18.65)**

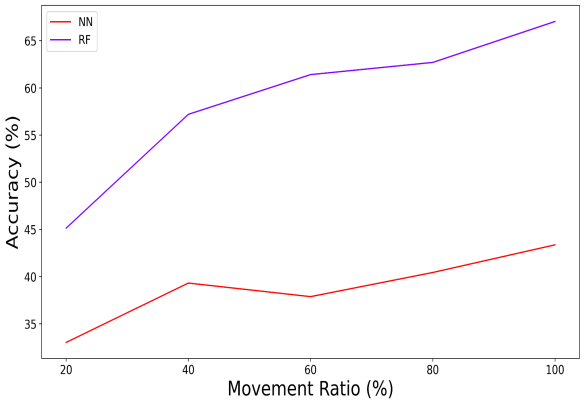

Figure 4.59: The accuracy rates of the Neural Net DL model and of the Random Forest ML model with respect to the movement ratio for the "one-out" dataset split and the 6th feature set.

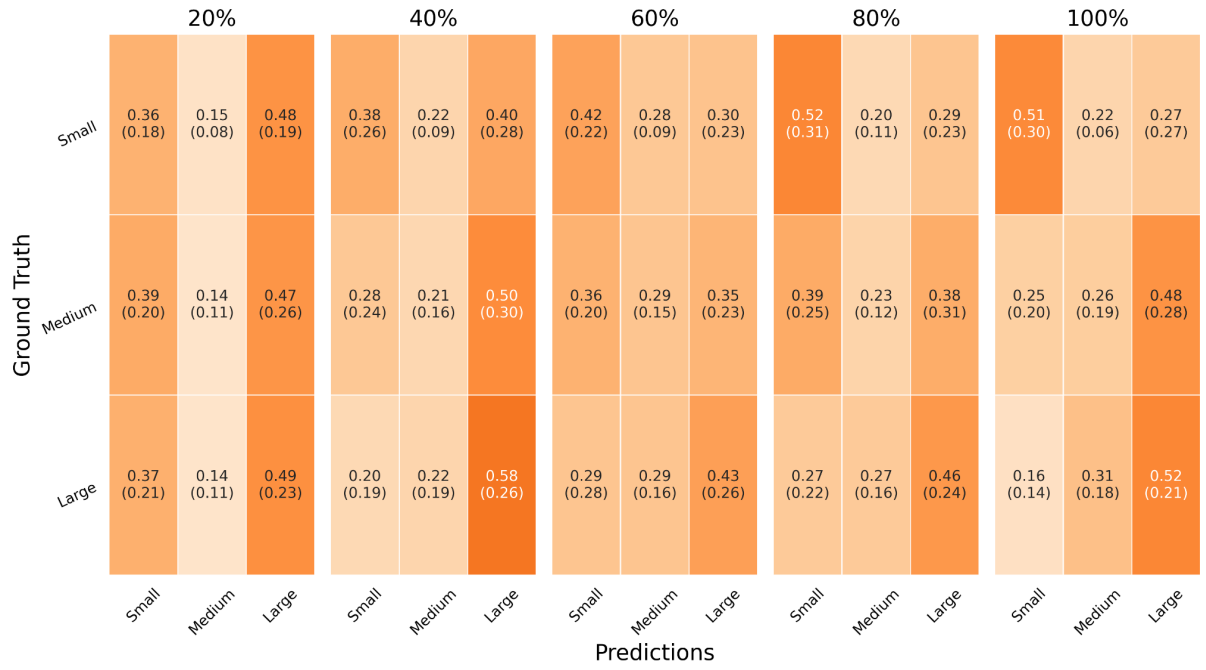

Figure 4.60: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 6th feature set with respect to the movement completion percentages.

For the **7th feature set** the results of the DL model are presented in Table 4.31 and Fig. 4.61. The 7th feature set contains the aperture and the wrist speed plane features (Table 3.1). The accuracy rates of the DL model were approximately between 30% and 45% for all the movement completion percentages. As a result, the accuracy rates of the Neural Net DL model were significantly lower than the accuracy rates of the top-performing Random Forest ML model for all the movement ratios. The poor performance of the DL model can be further analyzed based on the CMs of Fig. 4.62. The accuracy rates for the small, the medium and the large object were less than 52%, 29% and 58% for all the movement completion intervals.

Table 4.31: The results of the Neural Net DL model compared to the Random Forest ML model for the "one-out" dataset split and the 7th feature set.

	20%	40%	60%	80%	100%
NN	33.57 (5.90)	36.66 (11.82)	45.03 (16.38)	41.69 (11.03)	40.13 (8.71)
RF	**40.80 (9.38)**	**55.78 (8.17)**	**61.84 (11.81)**	**64.24 (18.60)**	**63.95 (18.44)**

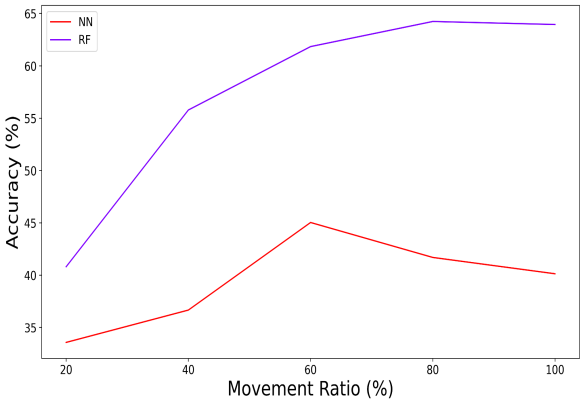

Figure 4.61: The accuracy rates of the Neural Net DL model and of the Random Forest ML model with respect to the movement ratio for the "one-out" dataset split and the 7th feature set.

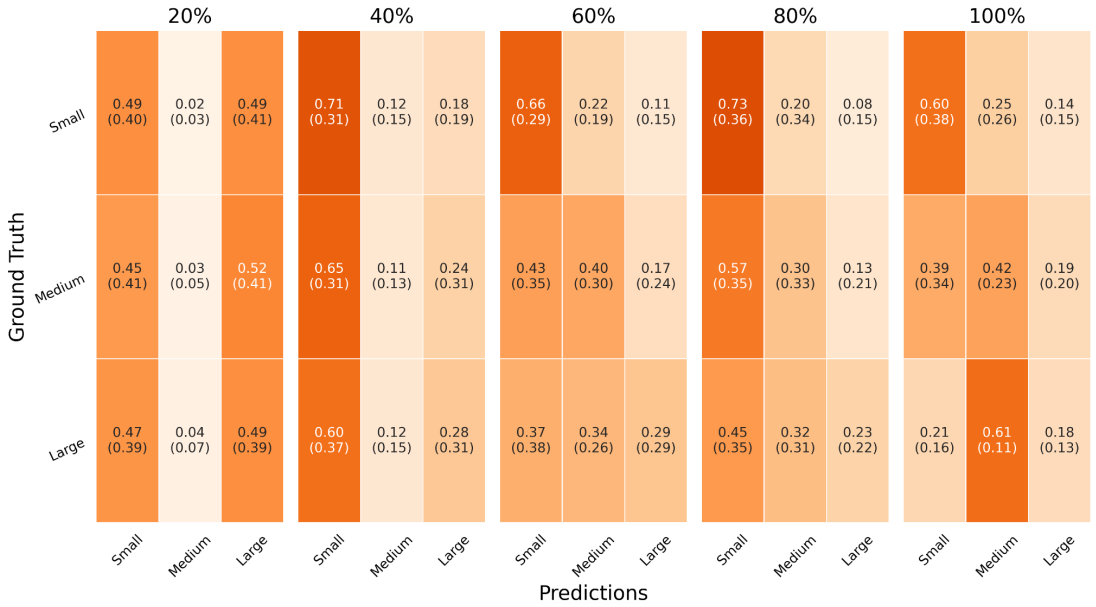

Figure 4.62: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 7th feature set with respect to the movement completion percentages.

The results of the **8th feature set** for the DL model are presented in Table 4.32 and Fig. 4.63. The 8th feature set contains the aperture, the wrist coordinate and the wrist speed axes features (Table 3.1). The accuracy rates of the DL model were between 30% and 40% for all the movement completion intervals. Compared to the top-performing Random Forest ML model, the accuracy rates of the DL model were significantly lower for all the movement ratios. Based on the CMs of the Fig. 4.64, the accuracy rates of the small, the medium and the large object were less than or equal to 50%, 38% and 42% respectively for all the movement completion percentages.

Table 4.32: The results of the Neural Net DL model compared to the Random Forest ML model for the "one-out" dataset split and the 8th feature set.

	20%	40%	60%	80%	100%
NN	33.99 (4.84)	32.45 (8.40)	36.78 (11.60)	37.79 (6.39)	37.08 (8.28)
RF	**41.64 (6.75)**	**56.22 (9.13)**	**59.04 (13.40)**	**63.10 (17.52)**	**64.79 (19.00)**

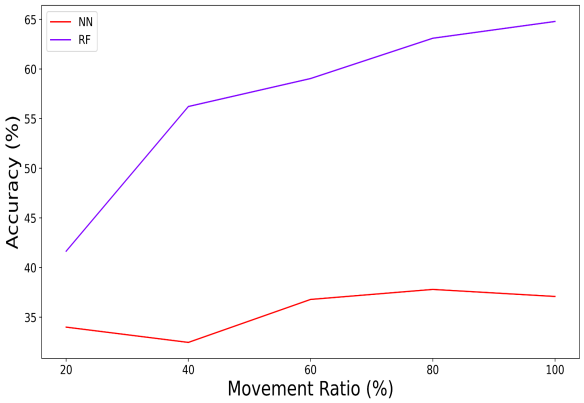

Figure 4.63: The accuracy rates of the Neural Net DL model and of the Random Forest ML model with respect to the movement ratio for the "one-out" dataset split and the 8th feature set.

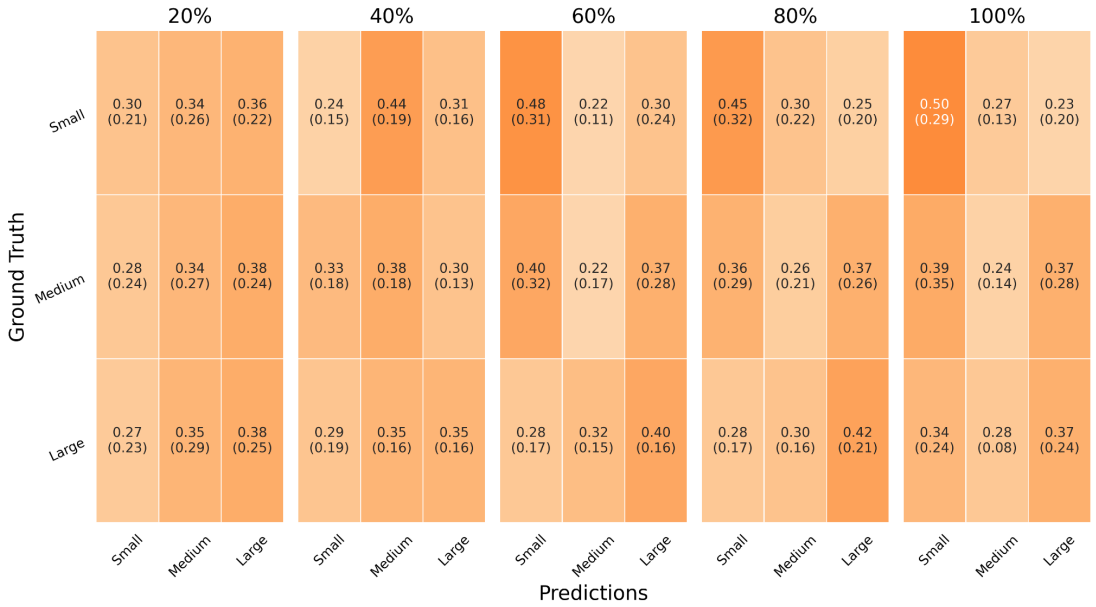

Figure 4.64: The confusion matrices of the Neural Net DL model for the "one-out" dataset split and the 8th feature set with respect to the movement completion percentages.

In conclusion, *the accuracy rates of the DL model were lower than the accuracy rates of the top-performing ML models for all the feature sets and all the movement completion intervals*. The best performance of the DL model was logged for the feature sets 1, 3 and 4. The **1st feature set** includes the aperture features, the **3rd feature set** includes the aperture and the wrist coordinates' dispersion axes features and the **4th feature set** includes the aperture and the wrist coordinates' dispersion plane features. The 3rd feature set is workspace-dependent, while the 1st and the 4th feature sets are workspace-independent. For the 1st feature set the accuracy rates were 38.85%, 46.82%, 48.38%, 55.83% and 64.47% for 20%, 40%, 60%, 80% and 100%. For the 3rd feature set the accuracy rates were 39.99%, 45.98%, 53.58%, 56.94% and 60.16% for 20%, 40%, 60%, 80% and 100%. Finally, for the 4th feature set the accuracy rates were 37.75%, 50.46%, 50.21%, 61.27% and 65.03% for 20%, 40%, 60%, 80% and 100% respectively. *In general, the standard deviations of the DL model were lower than the standard deviations of the top-performing ML models. However, since the margin*

between the accuracy rates of DL model and of the top-performing ML models was significantly large, these ML models were considered to be superior.

Moreover, based on the CMs of the DL model for the feature sets 1, 3 and 4 one can deduce similar conclusions. Specifically, after 80% of the movement the model learned to identify the small object as non-large and the large object as non-small with a maximum error rate of 10%. The main reason for the suboptimal performance of the DL model for these later stages of the movement are the small-labeled movements which were misclassified as medium, the medium-labeled movements which were misclassified as large and the large-labeled movements which were misclassified as medium. Finally, one can identify as a secondary cause of the suboptimal performance the medium-labeled movement which were misclassified as small.

4.2. Evaluation with respect to real-time robot performance

The results which were presented in the previous section of Chapter 4 were evaluated with respect to the real-time performance of a UR3 collaborative robot. The goal was to evaluate the latest movement completion interval for which the robot has enough time to respond to the human's movement. In our work, we considered a simple HRC scenario for which the robot must first predict the object on the table the human intends to grasp and reach it before the human. To this end, the data of Table 4.33 was used. This (unpublished) data was collected at the Roboskel[5] lab and provides information about the time needed for a robotic arm to cover a certain distance when moving at a speed of 0.9 m/s. Moreover, the data of Table 4.34 was used. This table contains information regarding the inference time of the top-performing ML models for different feature sets and for each of the movement completion percentages. Specifically, the inference time was calculated for the best "all-in" model (Extra Trees) and its best feature set (2). Similarly, the inference time was also calculated for the best "one-out" model (Support Vector Machine) and its best feature set (3). The time needed for the extraction of the meta-features for the ML models was included in the inference time of the model, since these meta-features cannot be calculated online for each frame (in contrast to the preprocessing and feature engineering that could potentially be altered to occur almost concurrently with the movement - more in Chapter 5). Instead the meta-features would be calculated after the kinematic features have been extracted for all the frames of the partial grasping movement. The inference times of Table 4.34 were computed in the context of the k-fold evaluation scheme. For each of the k dataset splits, the inference time was computed as the average of the testing set movements. Subsequently, these k average inference times were averaged to produce the values of the table. To simplify the process of calculating the needed response time, we considered the inference time equal to 0.01s for all the movement completion intervals.

Therefore, given a distance between the object and the robot, the needed response time was calculated as the sum of the corresponding movement duration and of the model's inference time. For example, for a distance of 0.03m, the needed response time was 0.098s, since the inference time and the robot movement duration were 0.01s and 0.088s respectively. Furthermore, the remaining time until the end of the grasping movement was calculated for all the frames based on the OpenPose timestamps of the current frame and the last frame of the grasping movement. Following that, it was determined whether the latest movement completion interval for which the robot could

[5] http://roboskel.iit.demokritos.gr/

respond on time was 20%, 40%, 60% or 80%. For each of the distances of Table 4.33, the results are presented in Fig. 4.65.

Table 4.33: The movement duration with respect to the covered distance for a robotic arm.

Distance (m)	Robot Movement Duration (s)
0.03	0.088
0.05	0.112
0.10	0.159
0.15	0.207

Table 4.34: The inference time (in seconds) of the top-performing ML models with respect to the movement completion intervals.

Model	Feature Set	Movement completion intervals				
		20%	40%	60%	80%	100%
Extra Trees	2	0.0079	0.0097	0.0085	0.0094	0.0092
Support Vector Machine	3	0.0013	0.0013	0.0013	0.0014	0.0016

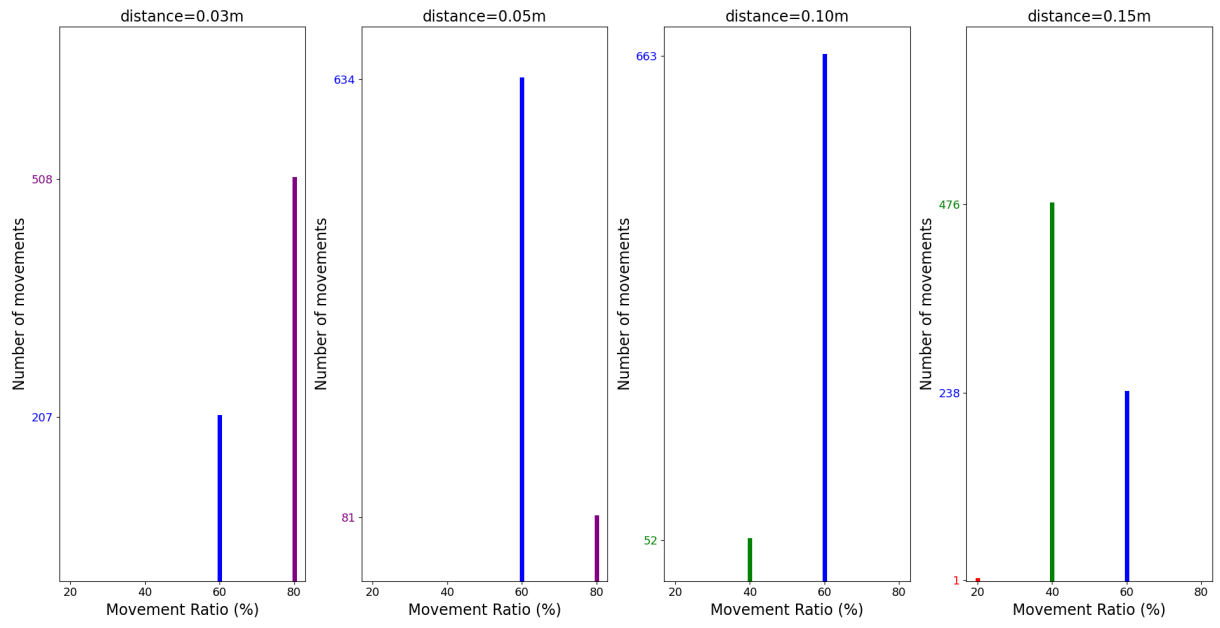

Figure 4.65: The distributions of the latest movement completion percentages for which the robot can respond on time with respect to the distance between the robot and the object.

Based on Fig. 4.65, the predictions' accuracy for the real-time HRC application was calculated with respect to the distance between the robot and the object. Both the "all-in" and the "one-out" dataset split strategies were considered. For the "all-in" dataset split, the accuracy was calculated for each of the distances as the weighted average of the accuracy rates of the Extra Trees model for feature set 2 (Table 4.2). Similarly, for the "one-out" dataset split, the accuracy was calculated for each of the distances as the weighted average of the accuracy rates of the Support Vector Machine model for feature set 3 (Table 4.19). For example, consider a scenario in which the robot is placed 0.03m away from the object. Based on Fig. 4.65, the robot would have to predict the object-to-be-grasped at 60% of 207 movements' completion time and at 80% of 508 movements' completion time in order to reach the object first. Furthermore, based on Table 4.2, the accuracy rate of the Extra Trees model for the 2nd feature set and the "all-in"" dataset split strategy was 86.01% and 91.88% for 60% and 80% respectively. Following this, the accuracy rate percentage (%) of the top-performing model for the "all-in" dataset split strategy and for 0.03m distance was calculated as:

$$86.01\% * \frac{207}{715} + 91.88\% * \frac{508}{715} = 90.18\%$$

The accuracy rates were calculated similarly for both dataset split strategies and for all the distances. The results are presented in Table 4.35.

Table 4.35: The predictions' accuracy rates (%) for a real time HRC application with respect to the dataset split strategy and distance between the robot and the object.

Dataset Split Strategy	Distance (m)			
	0.03	0.05	0.10	0.15
"all-in"	90.18	86.67	85.58	82.07
"one-out"	65.41	66.39	66.22	63.29

Based on this table, one can observe that for the "all-in" strategy the accuracy rates were larger than 80% for all the distances. Similarly, for the "one-out" strategy the predictions were approximately twice as good as random guessing, regardless of the distance.

5. DISCUSSION

Given the results which were presented in Chapter 4, one can compare the “all-in” with the “one-out” dataset split strategy, the “traditional” ML models with the DL model and the prediction accuracy rates for the small, the medium and the large object.

On the **dataset split strategies** comparison, *the performance for the “all-in” strategy was significantly superior than the performance for the “one-out” strategy*. This can be explained, since for the former dataset split strategy the models were trained with movements of all the participants. In this way, the model could predict the target object of a movement successfully even if the participant’s movement pattern was significantly different from the other participants’ movement patterns. On the other hand, for the “one-out” strategy, the models were trained with the movements of all but one of the participants. This participant’s movements were used for the cross-validation/testing set. As a result, in case the participant’s movement pattern was distinct from the other participants’ movement patterns, the model would not be able to generalize. The standard deviations of the top-performing models for the “one-out” strategy are evidence in this direction. Specifically, they were substantially larger than the standard deviations of the “all-in” strategy. The accuracy rates for participants which did not follow similar movement patterns with the rest were outliers and led to accuracy reduction and standard deviation increase. In order to achieve better results for the “one-out” strategy, collecting movement data from more participants would be helpful. In this case, it would be more probable for the movement pattern of a participant to be similar with the movement pattern of at least one other participant.

Regarding the **accuracy rates per target object**, one can observe that *the medium class was mainly responsible for the suboptimal performance both for the “all-in” and for the “one-out” dataset split strategies*. Specifically, either the small- and the large-labeled movements were misclassified as medium or the the medium-labeled movements were misclassified as small or large. In other words, *the model learned to identify the small object as non-large and the large object as non-small as the movement progressed*. Moreover, it is worth mentioning that the primary reason for the suboptimal performance of the models during the later stages of the movement was either the misidentification of the medium object as large or vice-versa. This can be explained since the small, the medium and the large objects were cubes with edge lengths 2.5cm, 5.5cm and 7.5cm respectively. As a result, *the medium object was slightly more similar to the large object and this minor difference was first reflected in the hand movements of the participants and thereafter in the predictions and the confusion matrices of the models.*

Finally, for the **“traditional” ML - DL comparison**, *the accuracy rates of the top-performing ML models were higher than the accuracy rates of the DL model for all the feature sets and all the movement completion intervals. This applied for both the “all-in” and the “one-out” dataset split strategy.* The main reason for this phenomenon was high variance. Specifically, after the DL model was trained, its accuracy on the training set was near-perfect, while the accuracy for the cross-validation/testing set was significantly lower. Figure 5.1 describes the steps which can be followed in order to train a DL model with low bias and variance. Steps 2, 3 and 4 were followed to lower the cross-validation error in this work. Specifically, as mentioned in section 3.5, the number of parameters of the model was reduced and the dropout regularization technique was applied with a high probability. Moreover, the problem was further mitigated when the feature sets which were used contained a smaller number of kinematic features. Nevertheless, despite the fact that the cross-validation error was reduced, it remained substantially higher than the training error.

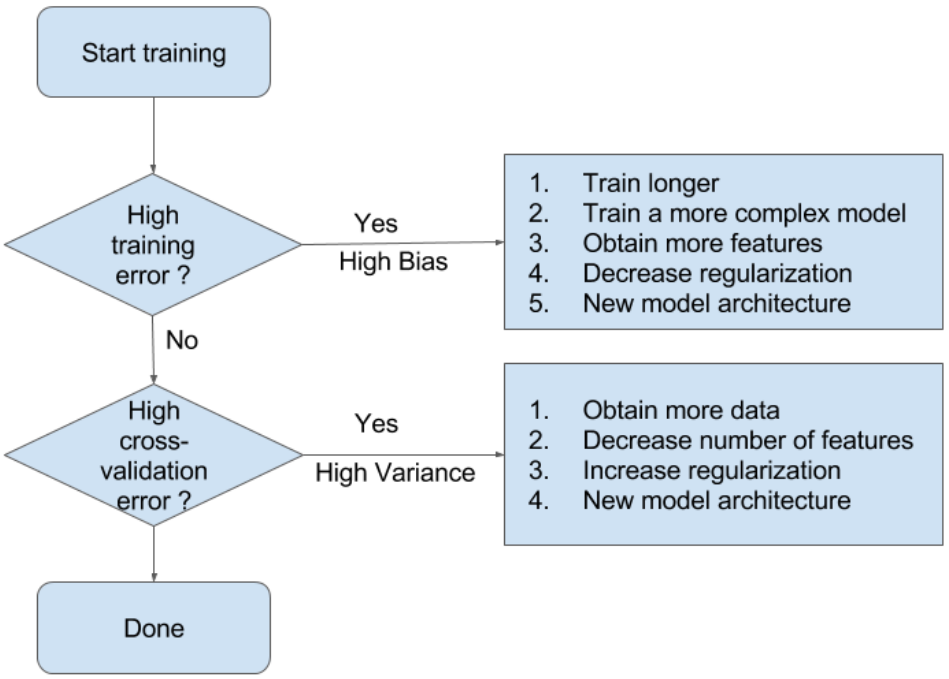

Figure 5.1: The steps to train a deep learning model with low bias and variance.

In order to reduce the cross-validation error even further, a solution would be the collection of more movement data. In general, the Neural Network models perform better than the “traditional” Machine Learning models when fed with a lot of data. Furthermore, another promising research direction would be to explore alternative model architectures. As mentioned in Chapter 2, Graph Convolutional DL models had great success in the task of skeleton-based action recognition. For this task the skeleton data of the full body was utilized to recognize the action which was performed by a human. Similarly to this approach, the skeleton data of the right hand can be utilized to predict the object to be grasped without first producing the kinematic features. The hand skeleton is naturally structured as a graph with the joints as vertices and the bones as edges. Therefore, a GCN model could exploit the graph structure of the hand data to fully express the dependency between correlated hand joints. Two GCN approaches which seem to be more suitable for our problem are [24] and [25]. Both of these approaches must be adjusted for a hand skeleton input representation instead of a full body skeleton input representation. In [24], the proposed approach achieves a state-of-the-art accuracy for the action recognition task with at least 10 times less computational complexity than other state-of-the-art models. Therefore, this light-weight architectural design could be utilized for fast, real-time predictions for our task. In [25] a novel regularization technique for GCNs, Adaptive DropGraph (ADG), is introduced, which may prove to be helpful for reducing the cross-validation error. For these reasons the aforementioned two approaches or a combination of them could be useful for the object prediction task.

In the previous chapter, the results of the ML models were evaluated with respect to real robot data. Specifically, a simple HRC scenario was considered. In this scenario, the robot had to predict the object in real-time during the grasping movement. Nevertheless, it was not within the scope of the methodologies which were described in Chapter 3 to classify a movement in a real-time, online manner and, therefore, the results were only

theoretically evaluated. The necessary changes needed for **a real-time application** are described subsequently.

Specifically, in order to extract the skeletal data from the RGB videos, OpenPose was used. The frame rate of the camera used to capture the RGB data was 60 FPS. To exploit all camera frames available each frame was processed off-line through OpenPose. However, *the real-time inference rate of OpenPose* is around 21.3 Hz for the Quadro RTX 4000 GPU. Therefore, for real-time extraction of the hand skeletal data at the same rate to that of the camera sampling rate, unless a hardware upgrade is possible, a different hand pose estimation algorithm must be used. Alternatively, one could train new models using a dataset for which the RGB information is processed by OpenPose in real-time. In this case, for the given frame rate and OpenPose inference rate approximately ⅔ of the frames would be discarded and, therefore, the task would be significantly more challenging.

Following the extraction of the skeleton data, the preprocessing pipeline was applied to each movement. For the filtering preprocessing step, the proposed methodology can be applied for a real-time application. However, for the grasping phase identification step, it is necessary for the skeletal data of the full movement to be available in order to identify the grasping movement's end. The same holds for the grasping movement completion intervals step. The intervals of each movement were identified given the grasping movement's end and, therefore, this step would not be applicable online during the movement. For a real-time application the object to be grasped would have to be predicted before the grasping movement's end. As a result, instead of a movement end criterion, a prediction time criterion would have to be devised. A simple prediction time criterion would be to infer the object after waiting a fixed amount of time from the beginning of the grasping movement. Given that the durations of the grasping movements differ from one another, this prediction time criterion may not be optimal for all the movements. Another approach would be to first approximate the peak of the wrist standard deviation and then to predict the object's size. For example, in order to identify whether the peak is observed in a frame t, one could consider its 3 previous and 3 next frames. In case the wrist's standard deviation for frame t is the largest compared to its 6 neighbouring frames, the model will predict the object. A disadvantage for this criterion is that it would not be applicable if the robot needs to respond to the human movement earlier than the peak of the wrist's standard deviation occurs.

Following the preprocessing pipeline, the kinematic features were engineered. In a real-time scenario, these kinematic features would have to be engineered in parallel with the preprocessing pipeline for each frame. Finally, both the "Traditional" Machine Learning and the Deep Learning methodologies can be applied for a real-time application after the kinematic features extraction. Specifically, for the "Traditional" Machine Learning methodology, the time axis values were calculated as the elapsed time since the beginning of the grasping movement and this information would be available in real-time. Furthermore, both the "Traditional" Machine Learning and the Deep Learning models which were examined can infer the object-to-be-grasped almost instantaneously.

ABBREVIATIONS - ACRONYMS

HRC	Human-Robot Collaboration
FPS	Frames Per Second
ML	Machine Learning
DL	Deep Learning
FLOPs	Floating Point Operations
ROS	Robot Operating System
GPU	Graphics Processing Unit
FLOPS	Floating Point Operations per Second
FS	Feature Set
CM	Confusion Matrix
RF	Random Forest
GB	Gradient Boosting
ET	Extra Trees
SVM	Support Vector Machine
GP	Gaussian Process
NN	Neural Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
GCN	Graph Convolutional Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory

REFERENCES

[1] B. Ren, M. Liu, R. Ding, and H. Liu, “A Survey on 3D Skeleton-Based Action Recognition Using Learning Method,” CoRR, vol. abs/2002.05907, 2020. [Online]. Available: https://arxiv.org/abs/2002.05907

[2] L. Wang, D. Q. Huynh, and P. Koniusz, “A Comparative Review of Recent Kinect-based Action Recognition Algorithms,” CoRR, vol. abs/1906.09955, 2019. [Online]. Available: http://arxiv.org/abs/1906.09955

[3] R. Vemulapalli, F. Arrate, and R. Chellappa, “Human Action Recognition by Representing 3D Skeletons as Points in a Lie Group,” in 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 588–595.

[4] B. Wu, A. Wan, X. Yue, P. H. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer, “Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions,” CoRR, vol. abs/1711.08141, 2017. [Online]. Available: http://arxiv.org/abs/1711.08141

[5] P. Koniusz, A. Cherian, and F. Porikli, “Tensor Representations via Kernel Linearization for Action Recognition from 3D Skeletons,” CoRR, vol. abs/1604.00239, 2016. [Online]. Available: http://arxiv.org/abs/1604.00239

[6] H. Wang and L. Wang, “Modeling Temporal Dynamics and Spatial Configurations of Actions Using Two-Stream Recurrent Neural Networks,” CoRR, vol. abs/1704.02581, 2017. [Online]. Available: http://arxiv.org/abs/1704.02581

[7] L. Li, W. Zheng, Z. Zhang, Y. Huang, and L. Wang, “Skeleton-Based Relational Modeling for Action Recognition,” CoRR, vol. abs/1805.02556, 2018. [Online]. Available: http://arxiv.org/abs/1805.02556

[8] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, “An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data,” CoRR, vol. abs/1611.06067, 2016. [Online]. Available: http://arxiv.org/abs/1611.06067

[9] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, “View Adaptive Neural Networks for High Performance Skeleton-based Human Action Recognition,” CoRR, vol. abs/1804.07453, 2018. [Online]. Available: http://arxiv.org/abs/1804.07453

[10] C. Si, Y. Jing, W. Wang, L. Wang, and T. Tan, “Skeleton-Based Action Recognition with Spatial Reasoning and Temporal Stack Learning,” CoRR, vol. abs/1805.02335, 2018. [Online]. Available: http://arxiv.org/abs/1805.02335

[11] B. Li, Y. Dai, X. Cheng, H. Chen, Y. Lin, and M. He, “Skeleton Based Action Recognition Using Translation-Scale Invariant Image Mapping and Multi-Scale Deep CNN,” CoRR, vol. abs/1704.05645, 2017. [Online]. Available: http://arxiv.org/abs/1704.05645

[12] C. Xie, C. Li, B. Zhang, C. Chen, J. Han, C. Zou, and J. Liu, “Memory Attention Networks for Skeleton-based Action Recognition,” CoRR, vol. abs/1804.08254, 2018. [Online]. Available: http://arxiv.org/abs/1804.08254

[13] C. Li, Q. Zhong, D. Xie, and S. Pu, “Co-occurrence Feature Learning from Skeleton Data for Action Recognition and Detection with Hierarchical Aggregation,” CoRR, vol. abs/1804.06055, 2018. [Online]. Available: http://arxiv.org/abs/1804.06055

[14] C. Cao, C. Lan, Y. Zhang, W. Zeng, H. Lu, and Y. Zhang, “Skeleton-Based Action Recognition With Gated Convolutional Neural Networks,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 11, pp. 3247–3257, 2019.

[15] Y. Li, R. Xia, X. Liu, and Q. Huang, “Learning Shape-Motion Representations from Geometric Algebra Spatio-Temporal Model for Skeleton-Based Action Recognition,” in 2019 IEEE International Conference on Multimedia and Expo (ICME), 2019, pp. 1066–1071.

[16] D. Liang, G. Fan, L. Guangfeng, W. Chen, X. Pan, and H. Zhu, “Three-Stream Convolutional Neural Network With Multi-Task and Ensemble Learning for 3D Action Recognition,” June 2019, pp. 934–940.

[17] S. Yan, Y. Xiong, and D. Lin, “Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition,” CoRR, vol. abs/1801.07455, 2018. [Online]. Available: http://arxiv.org/abs/1801.07455

[18] A. Shahroudy, J. Liu, T. Ng, and G. Wang, “NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis,” CoRR, vol. abs/1604.02808, 2016. [Online]. Available: http://arxiv.org/abs/1604.02808

[19] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Adaptive Spectral Graph Convolutional Networks for Skeleton-Based Action Recognition,” CoRR, vol. abs/1805.07694, 2018. [Online]. Available: http://arxiv.org/abs/1805.07694

[20] L. Shi, Y. Zhang, J. Cheng, and H. Lu, “Skeleton-Based Action Recognition With Multi-Stream Adaptive Graph Convolutional Networks,” IEEE Transactions on Image Processing, vol. 29, pp. 9532–9545, 2020.

[21] S. Yuan, Q. Ye, G. Garcia-Hernando, and T. Kim, “The 2017 Hands in the Million Challenge on 3D Hand Pose Estimation,” CoRR, vol. abs/1707.02237, 2017. [Online]. Available: http://arxiv.org/abs/1707.02237
[22] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, “catch22: CAnonical Time-series CHaracteristics,” CoRR, vol. abs/1901.10200, 2019. [Online]. Available: http://arxiv.org/abs/1901.10200
[23] A. Nielsen, “Modern Time Series Analysis,” presented at SciPy 2019 Conference, Austin, TX, USA, July 8, 2019. [Online]. Available: https://www.youtube.com/watch?v=v5ijNXvIC5A. [Accessed July 26, 2021].
[24] K. Cheng, Y. Zhang, X. He, W. Chen, J. Cheng, and H. Lu, “Skeleton-Based Action Recognition With Shift Graph Convolutional Network,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 180–189.
[25] K. Cheng, Y. Zhang, C. Cao, L. Shi, J. Cheng, and H. Lu, “Decoupling GCN with DropGraph Module for Skeleton-Based Action Recognition,” in Proceedings of the European Conference on Computer Vision (ECCV), 2020.
[26] B. Doosti, “Hand Pose Estimation: A Survey,” CoRR, vol. abs/1903.01013, 2019. [Online]. Available: http://arxiv.org/abs/1903.01013
[27] A. Armagan, G. Garcia-Hernando, S. Baek, S. Hampali, M. Rad, Z. Zhang, S. Xie, M. Chen, B. Zhang, F. Xiong, Y. Xiao, Z. Cao, J. Yuan, P. Ren, W. Huang, H. Sun, M. Hrúz, J. Kanis, Z. Krnoul, Q. Wan, S. Li, L. Yang, D. Lee, A. Yao, W. Zhou, S. Mei, Y. Liu, A. Spurr, U. Iqbal, P. Molchanov, P. Weinzaepfel, R. Brégier, G. Rogez, V. Lepetit, and T. Kim, “Measuring Generalisation to Unseen Viewpoints, Articulations, Shapes and Objects for 3D Hand Pose Estimation under Hand-Object Interaction,” CoRR, vol. abs/2003.13764, 2020. [Online]. Available: https://arxiv.org/abs/2003.13764
[28] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, “Real-time human pose recognition in parts from single depth images,” in CVPR 2011, 2011, pp. 1297–1304.
[29] J. S. Supancic III, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, “Depth-based hand pose estimation: methods, data, and challenges,” CoRR, vol. abs/1504.06378, 2015. [Online]. Available: http://arxiv.org/abs/1504.06378
[30] C. Wan, T. Probst, L. V. Gool, and A. Yao, “Dense 3D Regression for Hand Pose Estimation,” CoRR, vol. abs/1711.08996, 2017. [Online]. Available: http://arxiv.org/abs/1711.08996
[31] L. Ge, Z. Ren, and J. Yuan, “Point-to-Point Regression PointNet for 3D Hand Pose Estimation,” in Proceedings of the European Conference on Computer Vision (ECCV), September 2018.
[32] M. Rad, M. Oberweger, and V. Lepetit, “Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images,” CoRR, vol. abs/1712.03904, 2017. [Online]. Available: http://arxiv.org/abs/1712.03904
[33] Z. Zhang, S. Xie, M. Chen, and H. Zhu, “HandAugment: A Simple Data Augmentation Method for Depth-Based 3D Hand Pose Estimation,” CoRR, vol. abs/2001.00702, 2020. [Online]. Available: http://arxiv.org/abs/2001.00702
[34] S. Baek, K. I. Kim, and T. Kim, “Augmented Skeleton Space Transfer for Depth-based Hand Pose Estimation,” CoRR, vol. abs/1805.04497, 2018. [Online]. Available: http://arxiv.org/abs/1805.04497
[35] Y. Chen, Z. Tu, L. Ge, D. Zhang, R. Chen, and J. Yuan, “SO-HandNet: Self-Organizing Network for 3D Hand Pose Estimation With Semi-Supervised Learning,” in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 6960–6969.
[36] F. Xiong, B. Zhang, Y. Xiao, Z. Cao, T. Yu, J. T. Zhou, and J. Yuan, “A2J: Anchor-to-Joint Regression Network for 3D Articulated Pose Estimation from a Single Depth Image,” CoRR, vol. abs/1908.09999, 2019. [Online]. Available: http://arxiv.org/abs/1908.09999
[37] W. Huang, P. Ren, J. Wang, Q. Qi, and H. Sun, “AWR: Adaptive Weighting Regression for 3D Hand Pose Estimation,” CoRR, vol. abs/2007.09590, 2020. [Online]. Available: https://arxiv.org/abs/2007.09590
[38] Y. Cai, L. Ge, J. Cai, and J. Yuan, “Weakly-supervised 3D Hand Pose Estimation from Monocular RGB Images,” in Proceedings of the European Conference on Computer Vision (ECCV), September 2018.
[39] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, “GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB,” in Proceedings of Computer Vision and Pattern Recognition (CVPR), June 2018. [Online]. Available: https://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/
[40] A. Spurr, U. Iqbal, P. Molchanov, O. Hilliges, and J. Kautz, “Weakly Supervised 3D Hand Pose Estimation via Biomechanical Constraints,” CoRR, vol. abs/2003.09282, 2020. [Online]. Available: https://arxiv.org/abs/2003.09282

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, no. Oct, pp. 2825–2830, 2011.
[42] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, “Interactive, Collaborative Robots: Challenges and Opportunities,” in Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 18–25. [Online]. Available: https://doi.org/10.24963/ijcai.2018/3
[43] R. Chan, “How Learning From Past Industrial Revolutions Creates An Optimistic View Of Industry 4.0,” November 2019. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2019/11/01/how-learning-from-past-industrial-revolutions-creates-an-optimistic-view-of-industry-4-0/. [Accessed July 26, 2021].
[44] M. Guerry, S. Bieller, C. Müller, and W. Kraus, “World Robotics Report 2020,” International Federation of Robotics, Frankfurt, Hessen, Germany, September 24, 2020. [Online]. Available: https://ifr.org/downloads/press2018/Presentation_WR_2020.pdf. [Accessed July 26, 2021].
[45] V. Villani, F. Pini, F. Leali, and C. Secchi, “Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications,” Mechatronics, vol. 55, pp. 248–266, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957415818300321
[46] P. A. Lasota, T. Song, and J. A. Shah, A Survey of Methods for Safe Human-Robot Interaction, 2017.
[47] B. Rossi, “What will Industry 5.0 mean for manufacturing?” March 2018. [Online]. Available: https://www.raconteur.net/manufacturing/manufacturing-gets-personal-industry-5-0/. [Accessed July 26, 2021].
[48] C. Ansuini, A. Cavallo, A. Koul, M. Jacono, Y. Yang, and C. Becchio, “Predicting Object Size from Hand Kinematics: A Temporal Perspective,” PLOS ONE, vol. 10, no. 3, pp. 1–13, March 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0120432
[49] S. Han, “Efficient methods and hardware for deep learning,” Ph.D. dissertation, Stanford University, 2017.
[50] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,” CoRR, vol. abs/1812.08008, 2018. [Online]. Available: http://arxiv.org/abs/1812.08008
[51] G. Mpazakos, “Human action prediction based on hand movement for Human-Robot Collaboration,” Master’s thesis, University of Piraeus and National Centre for Scientific Research ’Demokritos’, 2021.
[52] P. M. Fitts, “The information capacity of the human motor system in controlling the amplitude of movement,” Journal of Experimental PSychology, vol. 74, pp. 381–391, 1954.
[53] Y. Kong and Y. Fu, “Human Action Recognition and Prediction: A Survey,” CoRR, vol. abs/1806.11230, 2018. [Online]. Available: http://arxiv.org/abs/1806.11230
[54] J. Romero, D. Tzionas, and M. J. Black, “Embodied Hands: Modeling and Capturing Hands and Bodies Together,” ACM Transactions on Graphics, (Proc. SIGGRAPH Asia), vol. 36, no. 6, pp. 245:1–245:17, Nov. 2017. [Online]. Available: http://doi.acm.org/10.1145/3130800.3130883
[55] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, “Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks,” ACM Trans. Graph., vol. 33, no. 5, Sep. 2014. [Online]. Available: https://doi.org/10.1145/2629500
[56] D. Tang, H. J. Chang, A. Tejani, and T.-K. Kim, “Latent Regression Forest: Structured Estimation of 3D Articulated Hand Posture,” in 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3786–3793.
[57] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, “Cascaded hand pose regression,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 824–832.
[58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An Imperative Style, High-Performance Deep Learning Library,” in Advances in Neural Information Processing Systems 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[59] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, “ROS: an open-source Robot Operating System,” vol. 3, January 2009.
[60] M. Dagioglou, A. C. Tsitos, A. Smarnakis, and V. Karkaletsis, “Smoothing of Human Movements Recorded by a Single RGB-D Camera for Robot Demonstrations,” in The 14th PErvasive Technologies Related to Assistive Environments Conference, ser. PETRA 2021. New York, NY,

USA: Association for Computing Machinery, 2021, p. 496–501. [Online]. Available: https://doi.org/10.1145/3453892.3461627

[61] Y. Kong, S. Gao, B. Sun, and Y. Fu, “Action Prediction From Videos via Memorizing Hard-to-Predict Samples,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/12324

[62] S. Ma, L. Sigal, and S. Sclaroff, “Learning Activity Progression in LSTMs for Activity Detection and Early Detection,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[63] Y. Kong, Z. Tao, and Y. Fu, “Deep Sequential Context Networks for Action Prediction,” July 2017.

[64] Y. Kong and Y. Fu, “Max-Margin Action Prediction Machine,” IEEE transactions on pattern analysis and machine intelligence, vol. 38, October 2015.

[65] S. Lai, W. S. Zheng, J. F. Hu, and J. Zhang, “Global-Local Temporal Saliency Action Prediction,” IEEE Transactions on Image Processing, vol. 27, no. 5, pp. 2272–2285, 2018.

[66] T. Lan, T. C. Chen, and S. Savarese, “A Hierarchical Representation for Future Action Prediction,” in Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 689–704.

[67] E. Coupeté, F. Moutarde, S. Manitsaris, and O. Hugues, “Recognition of technical gestures for human-robot collaboration in factories,” in ACHI 2016, 2016.