AI jury-assistent voor het herkennen van rope skipping skills in videos.

Mike De Decker.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele Bachelor toegepaste informatica

Promotor: Ms. L. De Mol & Mr. T. Parmentier
Co-promotor: Mr. D. Plummer
Academiejaar: 2024–2025
Eerste examenperiode

Departement IT en Digitale Innovatie .

AI judge for recognition of jump rope skills in videos.

Mike De Decker.

Thesis submitted in fulfilment of the requirements for the degree of
Professionele Bachelor toegepaste informatica

Supervisor: Ms. L. De Mol & Mr. T. Parmentier
Co-Supervisor: Mr. D. Plummer
Academic year: 2024–2025
First exam period

School of IT and Digital Innovation .

Preface

As a passionate jump roper, student of applied informatics, I came to the idea of creating an AI skill recognition system based on earlier seen ideas of sign langue detection and emotion recognition. These examples planted a blooming seed which I've proudly grown during my thesis.
I want to thank my promotors Lena De Mol and Thomas Parmentier for the feedback on the paper and the process. I also want give a big thanks to my co-promotor Dylan Plummer, to aid this Belgian Machine Learning enthiousast into tips and tricks for computer vision, model exploration, concepts and so much more. I also want to thank Pieter Gibens en Sara De Boever to aid in all my judging questions. Another special mention is for Gymfed, Eva and Arne, allowing to use earlier competition recordings and for the permission to record freestyles myself from 'behind' the judging table. I also want to thank Hogent (BIB Lera) for the ability to rent cameras. During the thesis, I've had a lot of joy in extending my judging knowledge, coding, labeling, training models, inspecting results and creating the Vue3 web app displaying some statistics.
The thesis may have finished, but I've only just begun!

Samenvatting

Door de evolutie van de sport is het jureren van ropeskipping freestyles op hoog niveau moeilijk geworden. Zowel het aantal skills in de routine, alsook de snelheid waarmee ze worden uitgevoerd neemt toe. Dit is vooral te merken in Double Dutch freestyles. Daarom worden deze routines zowel live (creativiteit, variatie, muziekgebruik) als vertraagd (moeilijkheidsgraad) gejureerd. Ondanks het feit dat freestyles op halve snelheid worden herbekeken en hierdoor jureerfouten worden vermeden, merkt men dat er nog enig verschil zit op scores toegekend door juryleden. Door de toegenomen toegankelijkheid van kunstmatige intelligentie, voornamelijk neurale netwerken, werd de vraag gesteld of een AI juryassistent ontwikkeld kan worden die helpt een betere en objectievere score zou opleveren.
Dit onderzoek verkent de mogelijkheid tot het bouwen van zo een juryassistent, de benodigde technieken en uitdagingen. De huidige vorm van de juryassistent bestaat uit drie hoofdzakelijke delen. Het eerste deel gaat over het lokaliseren van springers in de opnames. Niet alle videos zoomen in op de springer of zijn net eerder statische opname. Dit deel is noodzakelijk om computationele overhead te beperken, daar springers soms minder dan een vijfde van het beeld in beslag nemen. De tweede groote stap is het splitsen van volledige routines in elke uitgevoerde skill. Dit wordt gedaan aangezien het onbegonnen werk zou zijn op om dit manueel te doen. Het derde deel omvat het herkennen van de gesprongen skill. Voor Double Dutch Freestyles betekent dit een combinatie van uitvoering door draaiers en springers. Door louter presentatieskills of moeilijk zichtbare skills te makeren als 'unknown' (e.g. wanneer een draaier tussen de springer en camera staat), wordt er verwacht dat het model aangeeft wanneer het niet zeker is. Voor het lokaliseren slaagde YOLOv11 er in om een mAP50 te behalen tussen de 93-95%, waarbij het succesvol publiek filterde van atleten, mits kleine foutjes. Hierdoor het Multiscale Vision Transformer model skills ingezoomde crops gebruiken om acties van elkaar te onderscheiden. Deze konden vervolgens herkend herkend worden hetzelfde MViT model of een doormiddel van een Swin Transformer. Het gemiddelde f1 macro gemiddelde van deze modellen lagen tussen de 49 en de 53 procent, door de lage representatie van minder vaak voorkomende skills. Immers lag de totale accuraatheid hoger, tussen de 89 en de 94 procent. Dit zorgde ervoor dat juryscores door het model konden toegewezen, deze lagen -28 tot -20 procent onder de score toegekend door juryleden. Verdere onderzoek is nodig om de accuraatheid van de architectuur te verhogen.

Abstract

Judging jump rope freestyle routines at the highest competitive level has become increasingly challenging due to the evolution of jump rope. Both the number of skills that are included in a routine as well as the speed with which these are executed keep increasing. This is particularly evident in so-called Double Dutch Freestyle routines, which is why assigning scores to these freestyles is done by a combination of live and delayed evaluation. The creativity of a routine (including its variation and musicality) is scored in real time but the assignment of the appropriate difficulty level is done based on a recording of the routine replayed at half speed right after it is performed. Even though this helps reduce errors in difficulty scoring, a certain variability in the assigned scores persists/can still be seen. With the increased accessibility of artificial intelligence, particularly neural networks, the question arises whether an AI judge or assistant can be developed to obtain a more accurate (objective) difficulty scoring.

This research explores the possibility and development of such an AI assistant, as well as the techniques and challenges required to obtain the desired level of objectivity. The current idea is divided into three sections. The first section will be localizing the jumpers in the field as most obtained recordings are not fully zoomed in or recorded using a static camera. As recorded jumpers sometimes take up less than a fifth of the recording, they can be cropped out sparing computational resources for the parts to come. The second part involves isolating skills from a routine into individual skills or subskills. This enables the assistant to not only label a single skill, but also dozens of skills performed sequentially without interference. Lastly, each segment can be assigned to its corresponding skill. For Double Dutch Freestyles this means the combined action of jumpers and turners resulting in a large possibility of unique combinations.

YOLOv11 is used to predict the locations of athletes, filtering spectators, reaching a mAP50 of 93 to 95%. Despite minor mistakes, videos can be cropped and isolated in a segmentation model. This model, the Multiscale Vision Transformer is also used for recognizing skills. Along with the Swin Transformer, it reached the highest accuracies on skill predictions. The macro average f1 accuracy ranged between 49 and 53%, because of inaccuracy for lower represented skills. After all, the overall accuracy was higher, reaching 89 to 94% accuracy. Predictable skills allowed for these models to assign scores on freestyles. With an average ranging between -28 and -20 percent, scores assigned by models are nearing towards those awarded by judges. Further research is required to increase the accuracy of the architecture.

Contents

List of Figures

List of Tables

List of listings

1 introduction

Jump rope is an evolving sport. Year after year, an increasing amount of high-level competitors are pushing the limits of jump rope. This results in new skills, new combinations, better physiques, better rope material, and faster movements. For the judges to keep up with the jumpers and to correctly assess scores to a routine, Double Dutch freestyles [1], one of the jump rope disciplines, are reviewed at half speed in International competitions or even at nationals in Belgium.
Head judges around the world question the best way to judge athletes correctly so as to give an accurate and objective ranking in national or international competitions. Many solutions have been provided: another judging rule set [2], splitting judge responsibility, replaying the routine at half speed ...
Head jurors wonder about the possibility of using modern technologies like artificial intelligence to improve assigned scores of judges on jump rope freestyles. This idea is supported by an increasing popularity of image recognition, more powerful computers, applications that recognize objects in images (Singh Gill et al., 2022), implementations detecting simple human actions (Luqman & Elalfy, 2022), examples of action recognition in other sports (Yin et al., 2024), the integration of AI in gymnastics judging since 2017, namely the Judging Support System (JSS) and the successful test of NextJump's AI speed-counter.

1.1. Problem Statement

Jump rope, like gymnastics or athletics, consists of many events/disciplines. Examples include Speed, Single Rope freestyles (SR - single / pair / team), Chinese Wheel (CW) or Double Dutch (DD - single / pair / triad). On competitions, athletes perform

[1] Two turners, with one or more jumpers

[2] The current rule set is internationally enforced and maintained by the International Jump Rope Union where local adaptions are possible, such as the Belgian adaption enforced by Gymfed, closely related to the international judging-rules.

different skills of a single subdiscipline within a timespan of 60-75 seconds which is called a freestyle [3].

As jumpers can turn the rope with many arm restrictions or in different body positions, a lot of combinations can be created, especially in Double Dutch Routines. In order to decide the victor on competitions, levels are assigned to each performed skill, along with additions or deductions for proper execution of skills, use of music, movement, entertainment or variation.

To accurately judge routines, each discipline has its own rules. Although there is some overlap, there are also important differences. Having many rules, exceptions, different disciplines, high level athletes or fast routines makes judging prone to human error, (Heiniger & Mercier, 2018). This is evident in the discipline named Double Dutch, where judges are allowed to review the routine at half speed, in order to give a more accurate score, corresponding to the performed routine. This results in the aim of this research to improving jump rope judging. After all, judges are only volunteers, possibly not athletes, and the trick system is extensive and very hard to learn for an outsider (e.g a jumper's parent). Difficulty in judging is not just the speed the skills are performed but knowing the hundreds of different movements a jumper might do within less than a second.

1.2. Research question

Following the problem of judges unable to keep up, mainly during double dutch single freestyles, results in the main research question of this paper: **"How can the objectivity and accuracy of difficulty scores assigned to jump rope freestyles be increased using recent advancements in machine learning technology?"**.

To narrow down the scope, this is will be researched for Double Dutch Single freestyles (DD3), already containing Chinese Wheel parts. Let's jump into it.

What are the challenges for judges today?

During competitions, judges watch the routine live or delayed to annotate the difficulty or creativity. Each judge then pays attention to their assigned part, e.g. movement, musicality or difficulty, all of which contribute to the total score of the freestyle. The main problem now is for judges to keep up with double dutch routines. To increase the accuracy, difficulty-certified judges [4] are already allowed to review freestyles at half speed in order to score them more accurately.

How is difficulty scored?

Performed skills each have a level, which are be written down when seen by the judge. Each level also has a numeric score that contributes to the total difficulty of the routine. Judges see and calculate/memorize the level of each skill, write it down, count the number of levels jumped and calculate the diff score.

[3]Double Dutch freestyles already contains some Chinese Wheel
[4]Those judges judging the difficulty of double dutch routines.

What are the skills and transitions that need to be recognized?
Having multiple events means having more variation and more possibilities as an athlete. This results in different skills performable during freestyles. A description of skills is required in order to know what is expected to be recognized.

Which modern technologies can be used to increase score accuracy of jump rope freestyles
As reviewing routines at half speed still has his limits, using a machine learning (an AI-model) could reduce time spent on judging routines. The idea would be a machine learning model recognizing (sub)skills and transitions in a double dutch single freestyles. (DD3)

Which data is available for the machine learning model to use?
Working with machine learning requires data which raises the question whether it's available. To recognize hundreds of skills, variations and transitions, lots of data is needed. Both individual and team freestyles are mostly recorded by clubs themselves or event organizers, some of which are available on social media. The task is to explore and gather as much as possible.

Which activity recognition examples can be used or altered as a base guidance?
While the gymnastics judge support system (JSS) is a great example for recognizing skills, it's based on sensory data. The lack of sensors and the availability of video material explores this area. Quick searches give examples of object recognition (Diwakar & Raj, 2022), detecting sign language (Bora et al., 2023) or activity recognition (e.g. the kinetics dataset - riding a bike, reading a book, playing an instrument (Kay et al., 2017)). These implementations can be used as a first guide. Those examples give the idea that data mostly seem more to be centered, which is not the case in jump rope videos, a solution needs to be found for that. The second problem is that freestyles can consist of more than fifty different skills, which takes a long time to cut manually.

When are predictions acceptable to potentially use on competitions?
Judges do make mistakes, just like the machine learning models, but we do need a baseline for acceptable results. Past competition scores can be used to define a target.

1.3. Research objective

The objective is to create a proof of concept (PoC), which would be a model recognizing the most common skills and transitions. This could mean omitting or just marking special combinations, longer double dutch switches or long time sequences of emptiness in general. Preferably, the PoC should be able to generalize uncommon combinations of turners and skills. An example of this would be the model having seen triple unders, which may not have been performed with a cartwheel yet, but when it happens, it should know what it is.

How much data is expected to increase the accuracy of the Judge?
The amount of videos will keep rising, but will the current amount be sufficient? If it's not enough, how much more would be expected and what about uncommon skills. Do we need to specifically record them? And what will really happen with new skills during competitions?

1.3.1. Additional questions

The proof of concept will probably raise a lot of questions as a byproduct such as:

- How can we use the AI-Judge to improve judges?
- What needs to change on a working model, to apply it on other judge-related sports such as gymnastics, synchronized swimming, figure skating ...
- Can judges verify the labels predicted by the model in order to use them as new training data?
- Would allowing judges to annotate execution result in better objectivity?

1.4. Structure of this bachelor thesis

With some general knowledge about jump rope and the objective of the thesis defined, further research, (label)definitions, model selection, and implementation can be performed.
Let's start by exploring earlier work in section 2, while slowly increasing the number of jump rope definitions.
Using this research, a methodology can be created in order to recognize double dutch skills. This will be clarified in section 3.
In the fourth section (4), you can read all about the proof of concept, the most important highlights and code implementations.
Finally, you can read the conclusion in section 5, where answers will be provided about the research questions.

2

State of the art

The research towards skill-recognition will be done in several steps. The first step involves examining challenges of judging and recent adaptations for judging routines. Next follows a brief overview of DD3-skills, to further specify and clarify the DD3 components. Skills, combinations and transitions will than be mapped in a skill-matrix, nearing close to binary computer output. Upon specifying these components, it is revealed that their are certain challenges for AI models in order to recognize these skills. Something is needed that can use video information to localize the jumper, sequence the video, and recognize the performed skill. These three main requirements are elaborated in computer vision literature. Finally some challenges remain such as group activity or unknown/unusual skills.

2.1. Challenges of judging

As introduced earlier, based on own experiences and statements of colleagues, the sport is evolving. These statements are supported by fellow athletes, or commentary from the IJRU world championship livestream day 1 (IJRU, 2023a) to day 8 (IJRU, 2023b). Speed records are slowly rising, see tables 2.1 or 2.2 [1], also quads or quints in single rope freestyles are becoming the norm, where 10 to 15 years ago, it was considered a wow factor. This is also the case for double dutch; more variations, more turner involvements, faster and longer skill-sequences etc. The evolution of the sport requires an update of the evaluation of it. And indeed over the past few years, a few changes have been implemented to address some of the challenges the rising complexity of routines poses for human AI judges, such as splitting responsibility, multiple judging panels, the adaptation of rules or allowing post-review.

[1](Laue, 1999), (Gymfed, 2017), (IJRU, 2024b), (IJRU, 2024a)

Year	World	Europe	Belgium	Usa	Hungary	Germany	China
1998							
1999	80		80				
2012							
2015							
2016	111	103				103	
2019	111		102	105.5			
2020	111		102	105.5			
2021	111		103.5	105.5			
2022	111		103.5	105.5			
2023	113		103.5	106			113
2024	113	108	103.5	106			113

Table 2.1: History of speed records males, (Laue, 1999), (Gymfed, 2017), (IJRU, 2024b), (IJRU, 2024a)

Year	World	Europe	Belgium	Usa	Hungary	Germany	China
1998	83				83		
1999							
2012			102				
2015	105	105			105		
2016	105	105	102		105		
2019	108.5	105	102	100.5	105		108.5
2020	108.5	105	102	100.5	105		108.5
2021	108.5	105	102	100.5	105		108.5
2022	108.5	105	102	100.5	105		108.5
2023	108.5	105	102	100.5	105		108.5
2024	108.5	105	102	100.5	105		108.5

Table 2.2: History of speed records females, (Laue, 1999), (Gymfed, 2017), (IJRU, 2024b), (IJRU, 2024a)

2.1.1. Splitting responsibility

With the current rules, 11 judges, supervised by a head judge, are divided in two main categories. Three of them watch and annotate how difficult a routine is. The other 8 judge the creativity of the routine. Creativity is further split into execution, entertainment, musicality and variation, each judged by two jurors [2]. This breakdown allows increased attention on different aspects of a routine, thus decreasing potential observation mistakes.

2.1.2. Multiple panels

Using two or more judge-panels, more freestyles can be evaluated at the same time. While one panel is watching a freestyle, the other can summarize and calculate the total score of the previous routine. Panels and routines are often arranged to judge the same category, e.g. juniors vs seniors, which decreases the effect of human differences between judges, such as how strict one is when evaluating, incorrect memorization of skill levels, fatigue, ...

2.1.3. Adapting the rules

In order to account for rising difficulty levels, variation, new skills, simplified judging, ... rule guidelines change. This can impact the way freestyles are evaluated, in ways of thinking, memorizing or calculating the level, score or deduction of a skill. One such example are the current rules for Double Dutch freestyles which uses 'snapshot' principle, which is a way of judging where you calculate a skill level right after a jumper has performed a skill, as if you take a picture. However, multiple rotations of the rope aren't 'visible' in a single imaginary image/snapshot. In other words, a snapshot is a small time interval incorporating multiple different aspects for a judge to take notice of. Using this principle requires a lot of thinking during the routine, requiring the post-review, to correctly assign difficulty levels to skill-combinations. These rules where applied during the seasons 23'-24' and 24'-25' and will be changed again.

2.1.4. Review at slower speed

In recent years, the video replay was introduced on high-level competitions [3] to review a double dutch freestyle at slower speed to accurately assign the performed skill-level. Even in slow motion, differences between assigned scores of judges exist when calculating the total level of skill, transition, turners and rotational speed.

2.1.5. Challenges summary

Incorporating all these things brought jump rope to where it is. To increase the accuracy of difficulty score assignments, we try to find improvements. One of these

[2] In case there are insufficient judges, some positions remain vacant
[3] High-level competitions contain world championships or local finals (e.g. Belgium)

is exploring automatic skill-recognition by using AI. When skills are recognizable by a program, they can be mapped to their corresponding level, contributing towards the end score. Knowing what's represented in an image or a video is called computer vision.

2.2. Jump rope skills introduction

Earlier we described the presence of multiple disciplines in jump rope. One particular discipline within jump rope that is highly complex and therefore challenging in terms of difficulty scoring is a Double Dutch Single Freestyle (DD3). This study will focus on this type of freestyle to explore how and in what way recent advancements in machine learning can facilitate and improve difficulty scoring.

DD3 consists of two turners and one jumper alternating ropes. Elements in double dutch are similar to single rope, but different at the same time. The jumper can execute skills, mainly powers [4], gymnastics or footwork. Meanwhile, turners can manipulate the rope by rotating multiple ropes underneath the athlete in a single jump, called multiple unders. Another turner involvement could be turner-skills, which are restrictions like crossing the arms, restricting one of your arms behind your back (EB), maybe both arms (TS) or under your leg. Turners can even perform footwork, powers or gymnastics themselves. [5] To judge double dutch, 'snapshots' are taken, then the corresponding level is given depending on the combination of turners, skills and rope-rotations.
Like any discipline, mistakes can happen for which points are deducted from the total score.
The following examples illustrate how scores are assigned for DD3 routines. Levels are only given when the jumper is performing a skill, these can be raised by turner involvements or skill modifiers.

- Powers
  - push-up - (= plank position - lvl 2)
  - split (2)
  - frog - (= handstand) (2)
  - swift/V-kick (3)
- Gymnastics
  - cartwheel (2)
  - kip (3)
  - salto (4)

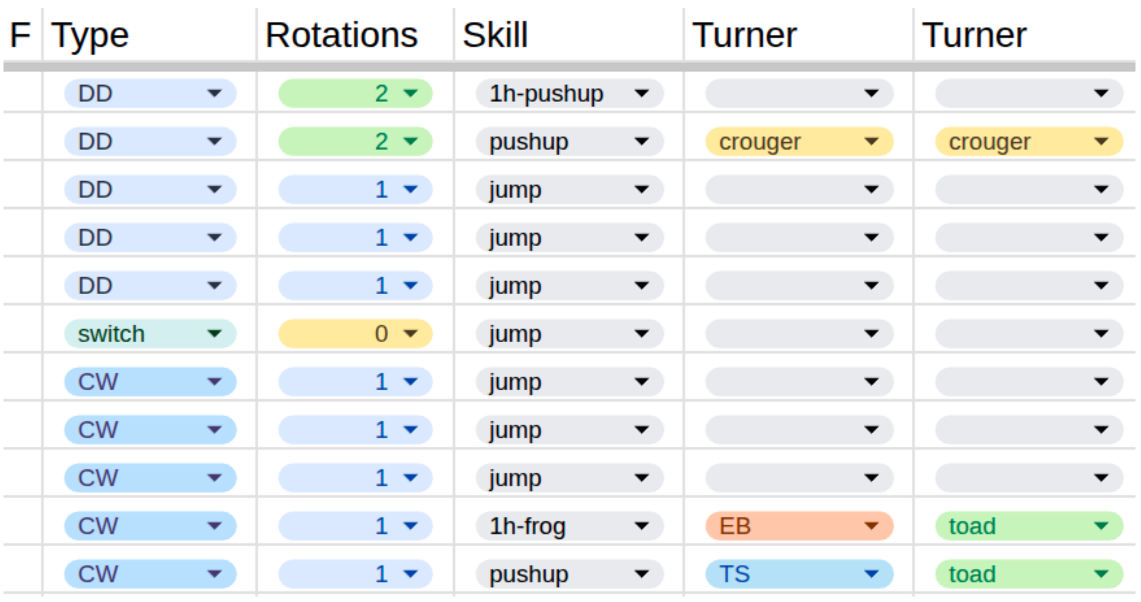
[4]powers/strength type skills
[5]Footwork as a turner doesn't raise your score.

- Turners
  - cross (c - crossed arms on the stomach) (+2/+0)
  - crouger (raise left knee, put left your arm underneath it from the inside to the outside) (+1)
  - EB (arm on stomach + arm on back) (+1)
  - TS (arms crossed behind the back) (+1/+1)
- Multiples
  - double - DU - 2 rotations (+1)
  - triple - TU - 3 rotations (+2)
  - quad - QU - 4 rotations (+2)
  - quint - 5 rotations (+3)
- Modifiers
  - using one hand (+1 or +0 depending on the skill)
  - pushing of 2 feet (+1 or +0 depending on the skill)
  - turntable - (+1 for each quarter rotation for a transition between the same power. e.g. push-up turntable = push-up to push-up with a quarter turn)

While these additions may seem simple, the combination of not knowing what will performed next, while keeping an eye on three people executing skills sequentially in about two thirds of a second makes it hard to evaluate for judges.

2.3. Skill-matrix - complexity & levels - towards model accuracy

The previous section, (2.2), provides a basic explanation of the skills to be judged, along some examples. To be able to develop a model that can accurately recognize all skills, it is necessary to draw up a detailed skills matrix. This section explains the composition of the skill-matrix, in order to give a better understanding on the total accuracy of the models later on. As described earlier, skills come with many transitions, take the first skill-matrix representation in figure 2.1 to better understand skills and transitions.
Type: You have four styles of turning in double dutch; the normal way or double dutch (DD), reversing the rope rotation, sort of like backwards called irish dutch (irish), the chinese way of turning, chinese wheel (CW) or only using one rope or the two combined as a single rope, called single dutch or one rope.
Rotations: The amount of ropes passing underneath the athlete in one jump.



F	Type	Rotations	Skill	Turner	Turner
	DD	2	1h-pushup		
	DD	2	pushup	crouger	crouger
	DD	1	jump		
	DD	1	jump		
	DD	1	jump		
	switch	0	jump		
	CW	1	jump		
	CW	1	jump		
	CW	1	jump		
	CW	1	1h-frog	EB	toad
	CW	1	pushup	TS	toad

Figure 2.1: First draft representing a skill-matrix used for Doube Dutch.

Turner: There are two turners, so two columns, where each turner can execute a turner involvement. Examples are EB, toad or crouger. The cross is in most cases performed by both turners, otherwise the ropes are tangled.
Skills: Mostly powers or gymnastics, but could also be footwork [6]. Further distinction or organizing can happen as variation and transitions of powers and gymnastics exists. A recap of different skills would be: pushups, splits, crabs, frogs, swift, cartwheel, salto, webster, suicide, handsprings, round offs ...Depending on the exact power or gymnastic, certain characteristics or transitions could be applied:

- one handed
- one or two feet push-off (e.g. frog vs high frog, salto vs webster, suicide one vs two legged push-off)
- turntable [7]
- full body rotation [8]
- consecutive [9], e.g. frog after frog

Applying these transitions, characteristics allows for even more variation, which will increase the number of awarded points. Repeated skills does not lead to. Repetitions are only defined by the skills in the ropes, the speed of the rope (rotations)

[6]Footwork will not be further specified in this paper
[7]On way of turning your bodyposition, can be done per quarter e.g. quarter turntable push-up
[8]Other way of turning your body, requires full turns
[9]Consecutive handstands are considered harder and thus get you extra points, not the case with push-up

	SR	DD3
#Freestyles	500-1000+	286-352
Hours	8h-16h+	5-6h
MVP	basic skill recognition	basic powers, gyms, turnerskills
Level-guessing	0 to 8	0 to 6/8
Theoretical level limit	8+ levels possible	8+ levels possible
skill-matrix	finer hand movements	larger actions compared to SR
individuals	1	3

Table 2.3: Data comparison between single rope freestyles and double dutch single freestyles. The number of freestyles is an estimation of the available and potential extra recordings possible on competitions during the research process.

and the type of turning [10].

The skill-matrix is subject to change over time. For the initial research and PoC, skills may be simplified or left out of the matrix will be left out for the PoC. Other than skills, jumpers and turners can switch positions, which is called a transition. These transitions do not fit in the matrix discussed below and are also omitted in the PoC. Each column in the first skill-matrix-example 2.1 can only contain one property, for which softmax can be used, while between multiple columns, no relation is required and meaning the skills can be predicted separately. This can be solved using a multi-branch output as described in by Coulibaly et al. (2022). This can result in a guessed skill being partially correct, e.g. turner correct, but wrong rotational amount.
On another note, Guo et al. (2017) explain that softmax isn't always a good indicator of confidence. They mention that when a model has good calibration, the accuracy should align with the confidence of the prediction. E.g. when you predict a skill has an 80% chance being of being a push-up, then this prediction or 80% of the predicted push-ups should be correct. Predictions using softmax tend to be overconfident.

2.4. Computer vision

To automatically recognize skills, input data is needed. There are quite some videos on socials, as well as in-house recordings, see table 2.3, so the choice to learn from videos was made quickly. This is also the choice for NextJump, a speed counter, counting how many speed-steps there are within a given time span. Computer vision is the field of study in which computers recognize features, people or other

[10]No difference will be made between irish turning and double dutch. Also, only the first skill in single dutch counts

objects in digital imagery. More specifically, the focus is recognizing human actions in these recordings, called Human Activity Recognition or HAR (Pareek & Thakkar, 2020).
Other papers mention slight adaptations to human action recognition, namely Human Gait Recognition, HGR, or Human Pose Estimation, HPE. Although the three techniques are closely related, each of them has a different nuance. Gait recognition looks at a person's typical movements, gestures or behavioral patterns (Alharthi et al., 2019), while pose estimation looks specifically at poses or special expressions (Song et al., 2021). They also talk about how pose recognition, e.g. skeleton-based, can be used as a tool to improve activity labeling. While these are interesting techniques, more effort is put into action recognition.

2.4.1. Computer vision in other sports

Soomro and Zamir (2014) published a book about the first advancements in computer vision since it was applied to sports. In addition, Yin et al. (2024) published a paper on the latest advancements of computer vision in teams competitions. Although relatively popular, neither of them really talks about gymnastics, which is closer related to jump rope than sports like tennis, basketball or cricket. Given examples by those two sources of performed computer vision tasks are: player tracking, following the ball trajectory (e.g. in cricket, football, tennis), human to human interaction (e.g basket) or detecting action types (e.g. running, walking, hitting the ball) or predicting the sport itself. (e.g. Olympics dataset).
A successful example recognizing static images on gymnastics rings, Abdullah and Alsaif (2023), illustrates the possibility of recognizing the main body positions in double dutch freestyles. Although the dataset is fully balanced and with a limited amount of classes, extending it shouldn't pose an issue. Preferably, identifying skills should incorporate the time aspect. Zahan et al. (2023) modified the Long Short Term Memory model (LSTM-model) to incorporate longer time sequences, to predict a full score of a routine which is static and not future proof. Changes in rules would make older scores useless and request for mistakes.
Meanwhile, the International Gymnastic Federation (FIG) has been working with Fujitsu since 2017 in order to create a Judging Support System. Initially, sensor data was used in order to predict skills, however in their official launch, (Fujitsu, 2023), they mentioned a transition towards camera based image analysis, reaching higher performance. The models have been tested and used on world championships, in order to review scores. The exact used model they use is unknown. Combining these examples and earlier research, a robust action recognition model can be developed detecting key aspects of a a video. These aspects, such as the number of hands, feet, body rotations and the main pose/action, can then be mapped with a level/score resulting in the full difficulty score of a routine.

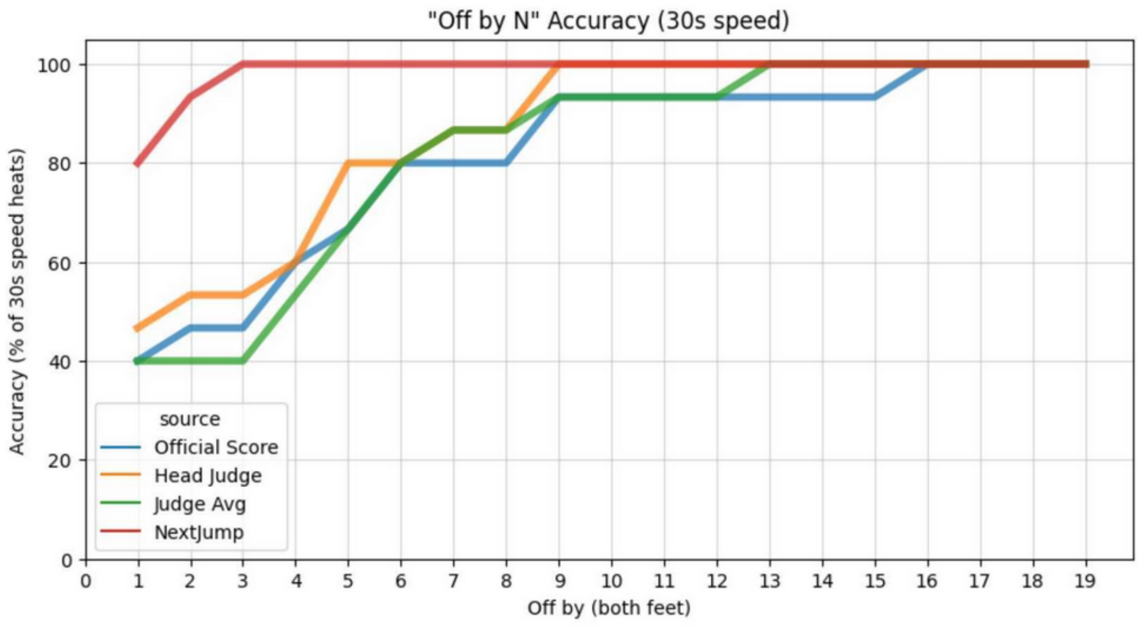


Figure 2.2: Comparison of avg scores given to a jumper, compared to the effective score. Results are from 2024 AMJRF nationals. Plot provided by NextJump

2.4.2. NextJump Speedcounter

As of august 2023, NextJump tested their AI-speed-counter, counting speed steps, on the world competition acquiring really accurate results, see fig 2.2. The speed counter, counts how many (right-footed) times the speed step is performed. A speed step is the alternation between jumping left or right footed, like running but stationary. Between each alternation, a rope goes under your feet. They found that 10 hours was sufficient for a single event (e.g. just single rope speed) but to count all kinds of events more (diverse) data is needed. The current dataset entails 36h video material, including other events such as triples unders or double unders. Using this as a base/guidance, the likelihood to succeed implementing skill-recognition in freestyles grows.

2.5. Human Action Recognition - general progress

Pareek and Thakkar (2020) summarized recent updates in human activity recognition. Based on their paper, a general approach is created in order to recognize the skills of jumpers.
As jumpers can stand everywhere in a field [11], locating and cropping athletes can improve the segmentation model [12]. When the skippers are centered, action segmentation can be performed. This allows for the predicting of skills on newly recorded videos, without the need to cut out the different skills. Finally, a level can

[11]A field is generally 12x12 or 15x15 meters
[12]If time allows it, the final model can be compared with or without localization

be mapped on each predicted skill [13]. Below you can find a summary of each step.

1. Jumper localization
2. Action segmentation, start/end of skill
3. Skill prediction
    (a) Predict skill (power/gymnestic - pushup, split, cartwheel, ...)
    (b) Predict turner involvement (cross, EB, TS)
    (c) Predict number of rotations (single, double, triple, ...)
    (d) Predict modifiers (1 hand, 2 feet, body rotations, ...)
4. Level & Score mapping

In what follows, a study of past and recent advancements are listed for each of the three steps, localization, segmentation and prediction. This allows for a selection of machine learning models, best suited for the task at hand. After selecting potential models, some potential hurdles are listed which can withhold progression on skill recognition.

2.6. Jumper localization

Image recognition has been at the center of many recent studies. The best models mostly utilize Convolutional Neural Networks processing spacial information in the image (Zaidi et al., 2021). Zaidi et al. (2021) further compare recent models for object detection such as YOLO(v4), CenterNet, SSD, EfficientDet-D2, each using some backbone architecture like VGG-16, AlexNet, GoogleNet or lightweight models such as ShuffleNet or MobileNet (all using CNN's). A subset of these models being real-time models (fps > 30). The goal of localizing the jumper is to center the athletes in the middle of the screen/video. Bharadiya (2023) states that the position of objects in images doesn't really matter, but there are no clear statements about the size of objects. It could be that a jumper takes up 80 percent of the screen, while moments later he moved backwards and only takes up 30 percent of the video. The assumption is that centered and scaled data will work better later.
For the purposes of this study, it is possible to use one of the recent models of image recognition as a starting point, using transfer learning[14] to fine-tune the results to localize the jumper.
To improve localization, video object segmentation or video instance segmentation can be used. VOS detects objects in videos, as compared to individual frames. Gao et al. (2022) lists some object segmentation models, like SwiftNet (H. Wang et al., 2021) which uses ResNet18 as good and quick model. Other possibilities would be Cutie (Cheng et al., 2023), DensePose (see fig 2.3) (Güler et al., 2018)

[13]Note that in skillrecognition can infer turner involvements, number of rotations, hands, feet, ...
[14]Concept transfer learning explained in (Bharadiya, 2023)

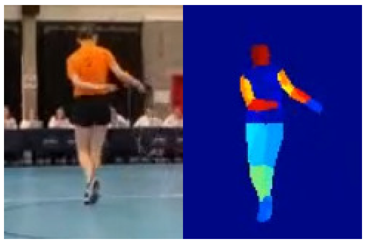


Figure 2.3: Jumper in a routine wrapping the rope around her arm in a single rope routine. Image on the left is the original image, on the right is the simplified information after using detectron2 densepose (Wu et al., 2019)

Densepose seems to be able to give the bounding boxes of the main poses detected. Perhaps just a convex hull and some padding will be enough for smart cropping and training a network from scratch is not needed. However, a local tryout (without boxes) was rather slow, 1.2 fps on GPU within a Docker container. As jumpers don't move that much most of the time, skipping some frames and smoothing out the prediction over the rest of the sequence or decreasing the amount of skipped frames when movement is detected can speed up the localization.
Even when Densepose isn't used, smoothing can still be applied to the guessed box as a video is basically a sequence of images.

2.7. Video action segmentation

When the athletes are cropped, videos need to be split [15] in (sub)skills, because splitting new videos manually takes too much time and is impractical on competitions. A more specialized model like LTContext (Zhou et al., 2023) or a video vision model explored in section 2.8 could suffice.
Just like in localization, extracting poses, filtering foreground from background etc. could improve the segmentation.
The model for generating skill snapshots will be useful for judging and subsequently labeling data. Rather than replaying the video, judges can just sequentially go through each trick one at a time to assign, annotate or validate the predicted skill.

2.8. Skill recognition

The final step would be to recognize the total skill in a freestyle video. Yin et al. (2024) did a general HAR survey, mainly focused on team sports, in which they describe the evolution from normal CNN architectures, to recurrent neural networks for time sequences, remembering context, like the Long Short Term Memory model (LSTM), after which models were developed feeding the CNN output into LSTM or even

[15] Not real/physical splits, but rather labels/annotations for where to split

implementing a convolutional filter in the memory cell (Shi et al., 2015).
Y. Wang et al. (2019) investigated an improvement for current convolutional or recurrent models. They found that LSTM memory cells were too simple to contain higher-order complexities. As a result, they designed a memory in memory component, to replace the previous cell, which could predict actions on more complex data sets. This quickly formed the name Memory in Memory, MIM. Later, Z. Lin et al. (2020) used a self-attention memory cell inside the convLSTM that can memorize global aspects in space and time. With about 35% the number of parameters compared to Wang's MIM-model, SAM achieved a similar score as on the moving MNIST dataset, but faster. Although the focus of these papers was predicting future actions, the output can be transformed into a classification model, rather than a prediction model.
Another approach would be using transformers as a described in Yin et al. (2024). Options are the video vision transformer (Arnab et al., 2021) or adaptions like the ViT-TAD model by Yang et al. (2023), Swin transformer Liu, Lin et al. (2021) or VideoMAE v2 by L. Wang et al. (2023). Further research/try-outs will be required to ensure the transfer models can predict actions.
For reference, NextJump uses a CNN - MobileNetv4, (Qin et al., 2024) and a transformer to analyze the full sequence and count. So using the convLSTM, SAM, MobileNet, or a transformer brings us to a better definition or example of what exactly we want to predict.
Even with a broad range of options, it isn't bad to think about potential hurdles. One of them are unknown of unusual skills. While a judge may refer to a head judge or think for itself, the AI model wouldn't be trained for that. The other hurdle could be that DD3 freestyles are a group activity, making it hard for models to recognize the required aspects.

2.9. Unknown/Unusual skills

Unknown skills or special cases pose another challenge. That's why the skill-matrix needs to defined in such a way, to accommodate for new combinations or skills. This is called zero-shot learning, (Pourpanah et al., 2022), where actions or combinations get recognized without any labels.
(Sort of marking unique skills as "I don't know" so that others new/unique ones will also be marked as "I don't know") Unusual skills on the other hand should be incorporated into the implementation. A more concrete example would be turntables, which are mainly performed using a crab or push-up, but also seen with a frog or a split. Turntables even have the potential to be combined with a swift. Omitting turntable frogs or splits in the train dataset can test this the ability to perform on unusual skills.

2.10. Summary literature

According to the state of the art, it appears feasible to analyze the performed skills in a given recording. In order to evaluate this, a proof of concept will be developed predicting jump rope skills. Sequentially, three steps are be required. The first step involves localizing the position of the athletes. This way, computational resources can be spared in the following steps, namely segmenting and recognizing. The segmentation consists of predicting split moments which should divide the video into likely skills. In the final step, the recognition step, each separated skill can be analyzed by a vision model, predicting the most likely skill performed for the given section.
When all skills are predicted, corresponding levels and numeric scores can mapped for each skill section. These scores add up to difficulty score, usable to decide a victor on competitions.

3 Methodology

This section contains a brief description of the road map to build the proof of concept (PoC), which combines the general key insights from the literature. Within the methodology a brief description is provided about each experiment, without specifying actual results or code implementations. These can be found in 4
The proposed model can be seen as a three-way modular setup, jumper localization, action segmentation and skill recognition. Each step can be improved individually in order to acquire the best possible result and is implemented independently. The first step focuses on localizing the athletes in the field in order to crop them out and spare computational resources. Next up is segmenting each skill performed in a given routine. The final part involves recognizing each isolated skill.
All experiments are performed using an Acer Nitro ANV15-51, running Ubuntu 24.04.2 LTS, using a 13th Gen Intel® Core™ i5-13420H × 12, with 16GB RAM and a NVIDIA GeForce RTX™ 4050 Laptop GPU 5898MiB.

3.1. Genral information

Before jumping into each different step, some general information is provided. Each of the 458 DD3 freestyles collected, 2231 total videos, has a record in the database. Two selections have been made to decide the videos in the validation set. The first selection is filtering all videos of this season (67 videos). The second selection uses the id of the video. In the beginning of the project, taking the 10th modulo of the id being 5 resulted in the most validation videos. Currently this corresponds to 42 validation videos. The other 90% are used as training videos (349). Each recording lasts for about 60 to 75 seconds, depending on the competition level and has around 40 to 60 performed skills. These do not include normal jumps. In order to annotate the videos, a custom Vue3 web application is created. In order to render the videos in the browsers, conversions from blu-ray (m2ts) or AVI

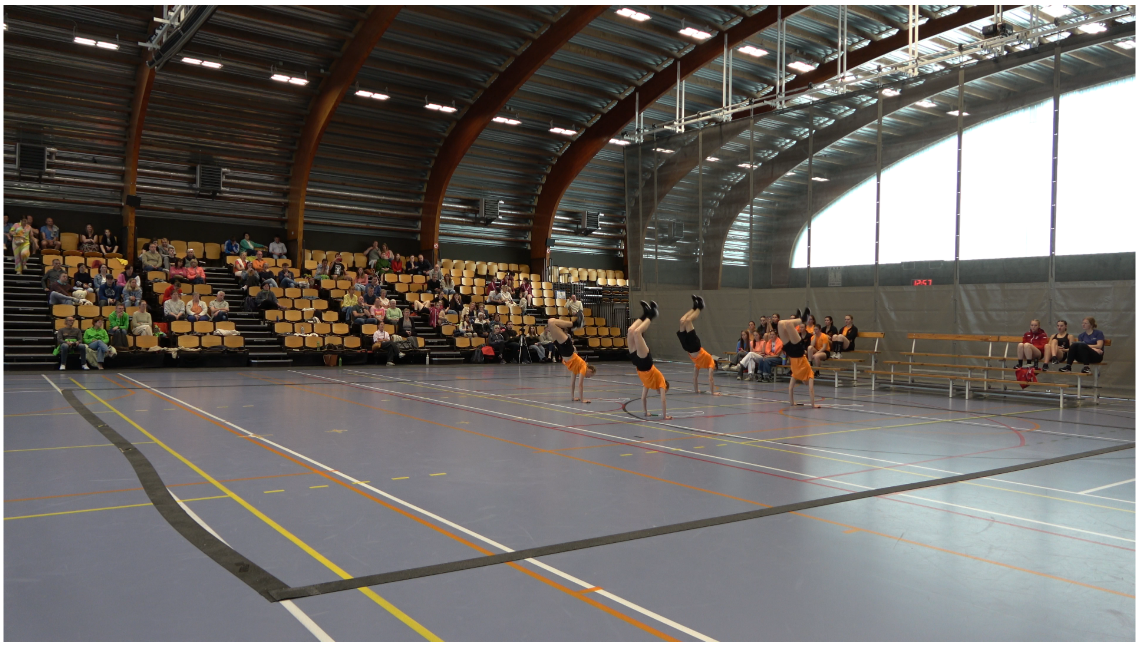


Figure 3.1: Example of a competition setting in which four athletes are performing a SR4 freestyle.

to mp4 were required. Most videos have a framerate of 25, 30 or 50 frames per second (fps), while some may be 15, 28 or 60. The quality of videos differ and range from anywhere between 8 and 550MB. It may not indicate much, but just to keep in mind. Even on the more qualitative videos, the rope disappears when making quick rotations. This is often better on higher fps videos.

3.2. Jumper localization

Jumper localization is needed in order crop a zoomed-in video of the skippers performing their routine. Image 3.1 perfectly illustrates a competition setting, where the majority of the video is lost on surroundings, judges and spectators. The field is typically 12 by 12 meters for the athletes to use.
Experiments on localizing implement two major ways of labeling athlete locations, full team boxes and individual boxes. For this project the relative center point along the x-axis, y-axis, width and height of the box are stored. So regardless of the scale of the image, whether the image has size 1920 x 1080 pixels or 1080 x 720 pixels, the position of the box remains the same. This way, by scaling all images to the same size, frames of dimension (width, height, channels) can fed into the model, giving 4 values in return. An example of a box would be [0.6, 0.5, 0.4, 0.4], all values range between 0 and 1. Team labels require one box for a single image, contrary to multiple boxes for individual athletes. After training and predicting box coordinates, the Jaccard similarity coefficient otherwise known as the Intersection over Union accuracy (IoU) can be performed evaluating the performance of the model. This allows for comparison of models or training rounds.

Full team predictions were experimented on MobileNetV3 (Howard et al., 2019) and GoogleNet (Szegedy et al., 2014). Labeling some frames of almost any DD3 routine, variety in the leftover data was limited. This marked the transition towards individual labels.
Ultralytics, (Khanam & Hussain, 2024), provides an easy way to implement their YOLO models. For this research, YOLOv11 is used to predict individual boxes, providing satisfying results 4.1.2. Using the predicted locations of the skippers, each frame of video can be cropped around the athletes. Results weren't perfect, sometimes visual disturbing, because of sudden zoom outs, blurry athletes or predictions of spectators. This required smoothing techniques to improve full video crops 4.1.2.
Manually reviewing full cropped videos is time consuming, requiring automated tests in order to check full video crops. The implementation of these automated tests used the full team boxes comparing the minimum IoU overlaps for each video, as these moments mattered most. By manually checking videos, problematic moments were additionally labeled as full team boxes. However, the further development and full video crop results were unfinished and put on hold due to time constraints.

3.3. Action segmentation

The main purpose of the action segmentation is to enable predictions on full routines in order to actually use the whole model. Using the cropped images, and the assigned start and end frame of a skill, the model can predict whether a given frame is an interesting split point or not.
For jump rope, this mostly means moments when an athlete leaves or lands on the floor. However there are special cases such as cartwheels. Only moments when either hands or feet land over a rope, are annotated as the end of a skill. To annotate which moments are interesting split points, the start and end frame of a skill receive value 1, while other frames remain 0. As the start and end frame of a skill are rather subjective, frames around a split point also range somewhere between 0 and 1. These values are calculated using the function C.6 which is called on in the data generator demonstrated in code C.7. Each frame than has a value from zero to one. An example of consecutive splitpointvalues could be:
$[0, 0, 0.1, 0.31, 0.55, 0.86, 1, 0.86, 0.55, 0.31, 0.1, 0]$
Having a splitpoint value for each frame of the video, the video can be partioned into consecutive sections of T timesteps. Because the Multiscale Vision Transformer is pre-trained on videosections of 16 frames, the same amount of timesteps are taken to finetune the model for segmentation. This makes the model transform the input of size (batch size, channels, timesteps, height, width) into an output of size (batch size, timesteps).
Available experiments include code implementations of the Video Vision Transformer (ViViT) as introduced in Arnab et al. (2021) and the Self-Attention ConvLSTM

from Z. Lin et al. (2020). Both models showed the same results, averaging out predictions slightly above 0 for all frames, stagnating on MSE losses, not actually grasping what happens in given sections of frames it receives.
The final experiment on action segmentation uses PyTorch's Video MViT, which is an implementation based on the Multiscale Vision Transformer created by Fan et al. (2021) which can be finetuned using the pre-trained weights on the kinetics dataset (Kay et al., 2017). This experiment will be elaborated in section 4.2.2.

3.4. Skill recognition

The last part involves recognizing each performed skill, which is, as introduced in the literature, a combination of different aspects. What are the turners doing, using one or both arms, an additional body rotation, etc. For this part, a video vision model is definitely required to fill in the temporal information, e.g. amount of rotations.
One minor issue, is about skills differing in duration. One might take half a second, the other might be 800 milliseconds. This can be solved by ensuring they have the same length of T = 16 timesteps, skipping or duplicating frames. This resulted in equal sized inputs with dimensions (channels, timesteps, height, width) or (C, T, H, W) for short.
Next up was transforming all aspects of skills into machine understandable output labels. This was the reason why the skill matrix has been created 2.3. The matrix resulted in a skill configuration C.8 for skill labels, specifying 13 different outputs; Type, Rotations, Turner1, Turner2, Skill, Hands, Feet, Turntable, BodyRotations, Backwards, Sloppy, Hard2see and Fault. Each aspect being a category, numeric value or a boolean.
Along the way, additional aspects to a skill label can be added in order to incorporate for every level and skill variation. Furthermore, once-in-a-lifetime skills can be marked as unknown in order for the model to be robust in predicting new skills. After analyzing the given skillsection, a level can be assigned, based on the predicted aspects. Multiple different models can be tested in order to find the best one.
The first experiments implement the Video Vision Transformer and the Self-Attention ConvLSTM, which averaged out the most occurring aspects; normal jumps, no mistakes, 1 rotation... These models were trained on upsampeled data, increasing occurrences of more unique skill combinations.
Following unsuccessful trials trained from scratch, accelerated the implementation of the Multiscale Vision Transformer for recognizing skills. At the same time, in order to test things out, skills where limited to to a balanced 5 class dataset; jumps, return from powers, pushups, frogs and other skills, hoping for better results. The results where excellent, even starting to recognize aspects of turners, quickly re-enabling predictions on all skills 4.3.2.
Further experiments added other models, namely video ResNet models or Swin

Transformers 4.4 or adjusted the returned losses, based on skill aspect occurrence in the dataset during training and validation 4.3.4.

3.5. Judge score comparison

Finally, when skills are predictable, a comparison between the jury assigned scores, the models prediction and the effective score can be made in order to check whether model predictions could be used or more research/training is needed. This is an important check, verifying usability on competitions 4.3.3.

4
Results

The result section focuses on elaborating conducted experiments and acquired results. It specifies actual results, details about implementations and code snippet references.

4.1. Jumper localization

As DD3 routines are the focus of this research, the idea was to predict the coordinates and position of a team as a single unit, instead of individuals. This idea was supported by the fact that there aren't any public jump rope datasets and it was expected to increase the speed of labeling. Image 3.1 shows an example of a competition setting where four athletes are performing a routine. The goal here would be to crop out the jumpers.
Below you can find a shortlist of experiments executed on localization.

- GoogleNet
- MobileNet
- YOLOv11

4.1.1. Dedicated architectures

This section explains the experiments using dedicated architectures. Results weren't good on about 6200 train and about 800 validation images. One issue might be that these models were finetuned for classification and not specialized for localization. A second issue might be the low amount of labels. Thirdly, predicting full boxes may not have been possible at all for the low amount of labels. The first try-out used a personal implementation, based on school courses of the GoogleNet architecture; (Szegedy et al., 2014) - C.3. It used the an input shape of

(Width, Height, Channels) to predict a full box around all three jumpers. Just like the implementation using MobileNetV3, (Howard et al., 2019) - C.4.
Predictions only shifted a little towards the athletes, averaging out in the center of the image, reaching an IoU of about 28 to 30%. These results where performed on the same dataset as in the experiment above.

- MobileNet IoU unknown / lost
- GoogleNet IoU 31.06%

Predicting full teams didn't work out which allowed for a transition to labeling and predicting individuals athletes. Below you find more information about those experiments.

4.1.2. YOLO

Experiment 3, pre-trained models & individual boxes.
The third experiment incorporates a double upgrade. The first upgrade involves annotating individual athletes on images instead of one big box around all jumpers. This method allows to annotate other routines such as single rope, chinese wheel or double dutch 4. The second upgrade is trying a pre-trained object detection model.
Ultralytics (Khanam & Hussain, 2024) provides an easy-to-use pre-trained implementation for predicting people and objects in images which can be fine-tuned for specific use cases. Fine-tuning was needed as spectators, also humans, were also included in the predictions.
Using these predictions, a crop around the three jumpers can be created using the predicted jumper boxes from the fine-tuned YOLO model, after using the pre-trained weights on the COCO dataset (T.-Y. Lin et al., 2014). In order to improve the crops some steps were required. Details are discussed below.

Crop stability

Annotating only athletes on 732 images & fine-tuning the pre-trained nano yolo model, the first predictions of spectators are reduced considerately. This is illustrated in figure 4.1. This way crops can be created by taking the minimum and maximum x & y-coordinates of all boxes predicted on a frame. Predicting these crops for all frames, i.e. a full video crop. However, reviewing cropped videos indicate occasional frames predicting spectators, coaches or judges and moments of instability, as if you are watching camera footage of an earthquake.
In order to avoid a sudden zoom out, because of spectators, an IoU comparison of the previous predicted box with the new box is made. If the IoU value is larger than the average IoU of the last N seconds, powered to the fourth, than crop coordinates will be updated. Using this IoU comparison, reduces video crops including spectators, maintaining the earthquake effect.

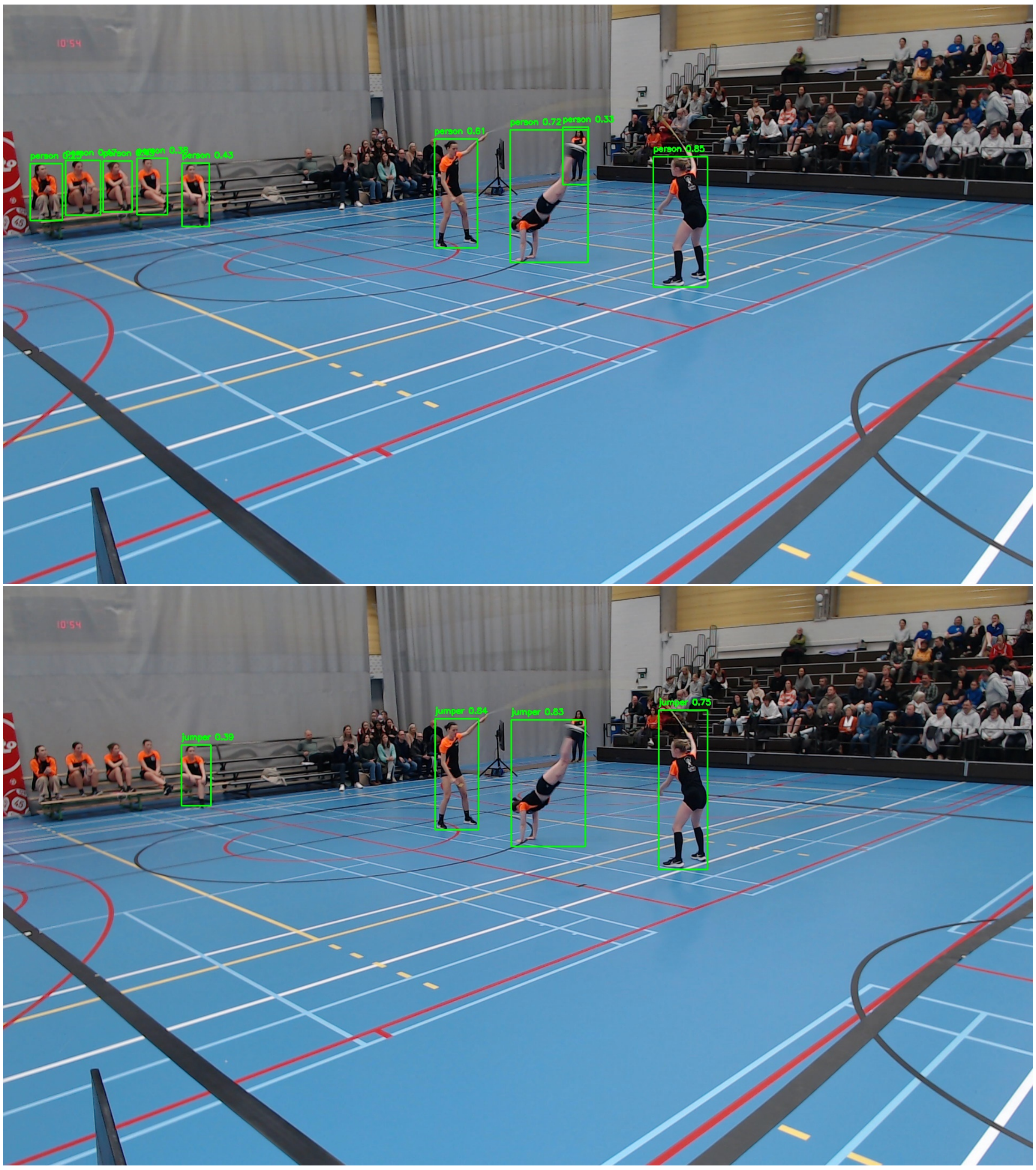


Figure 4.1: Raw predictions of the unrefined YOLOv11 nano model compared to the fine-tuned model which reduces predictions of spectators.

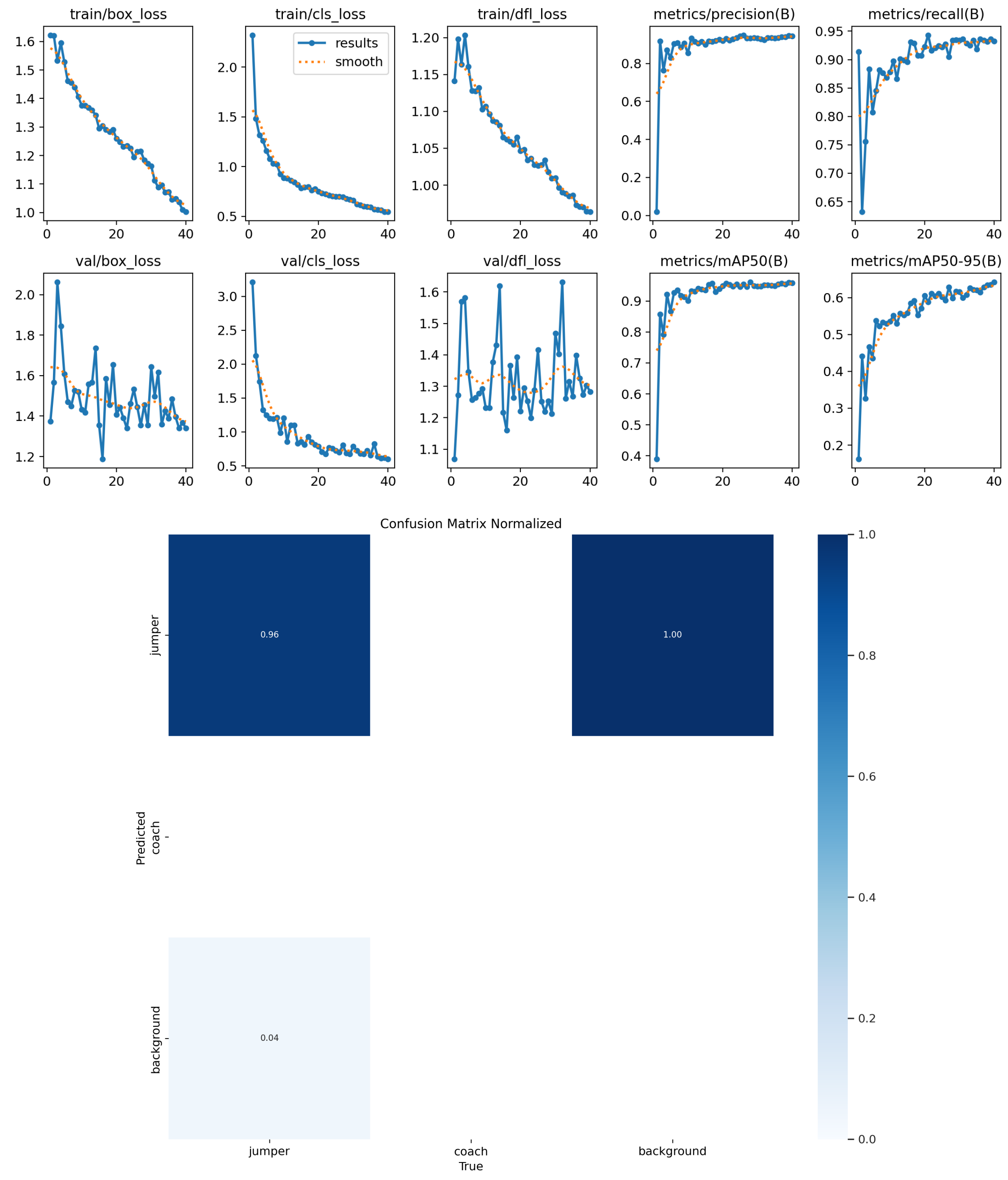


Figure 4.2: Metrics after fine-tuning YOLOv11 on the validation set of 161 images, mAP50 of 0.953 & mAP50-95 of 0.63, annotation of coaches was an idea.

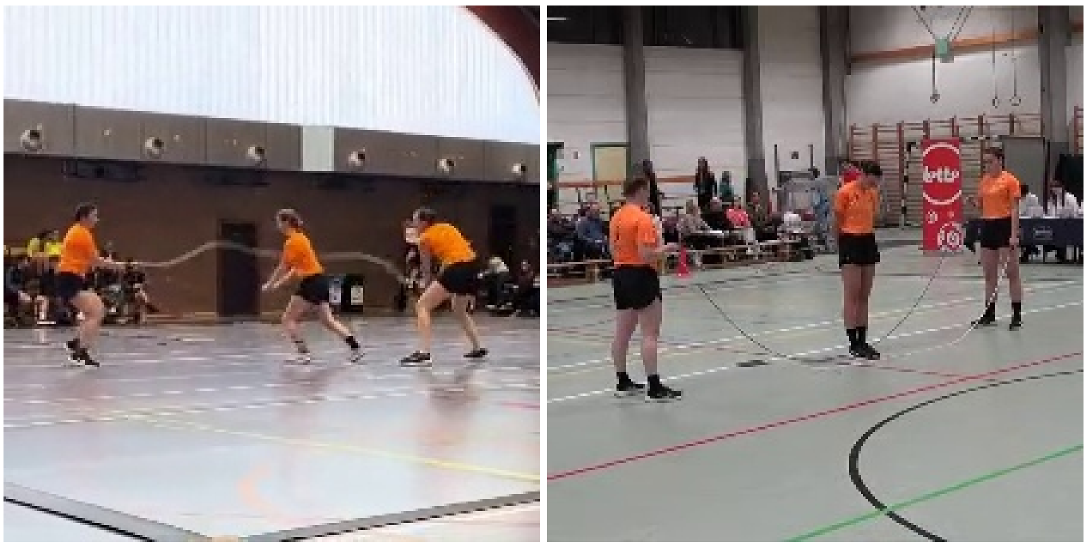


Figure 4.3: Valid crops of a DD3 routine on competition. (224x224 pixels)

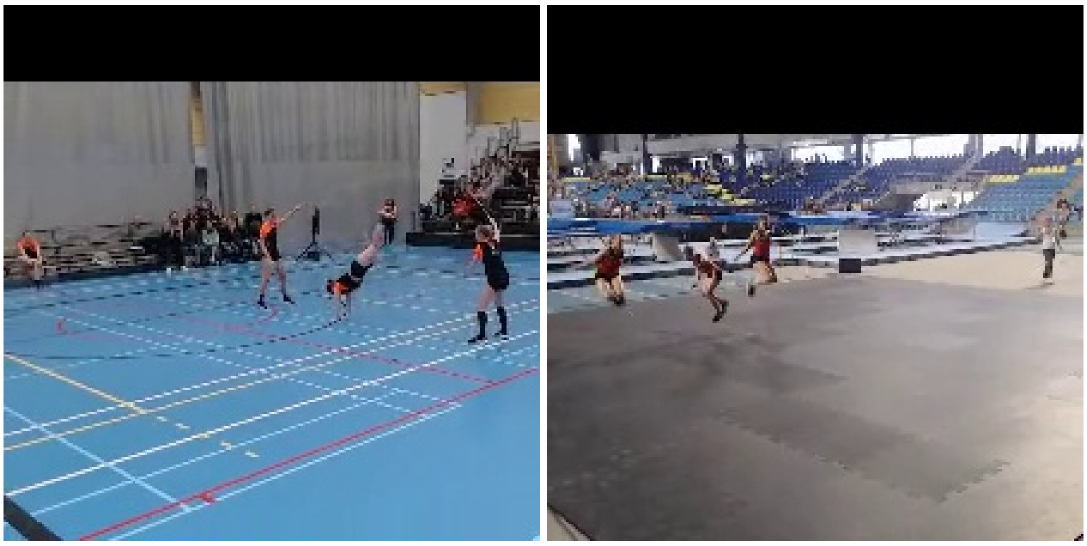


Figure 4.4: Two crops including a spectator alongside the athletes.

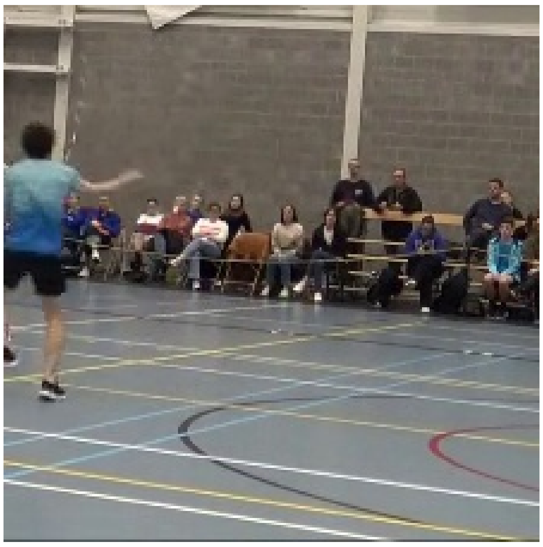


Figure 4.5: Athletes running out of view.

To reduce shakiness and maintain a nice to watch cropped video, box-coordinates are smoothed out adapting box predictions of previous frames with the current one. This way, drastic changes in a predicted location, e.g. model error, arm or field movements aren't that drastic. Adaptation is done by incorporating two smooth values, indicating how much weight you give to the smoothed prediction of the previous frame. [1]

$$smth = smootval = 0.86 \quad \& \quad smths = smootval\_shrink = 0.94$$

Which makes the new coordinates:

$$smoothed_{x1_{min}} = \begin{cases} smooted_{x1_{min}} & \text{if} \quad iou < iou_{threshold} \\ smth * smooted_{x1_{min}} + (1 - smth) * x1_{min} & \text{else if} \quad x1_{min} < smooted_{x1_{min}} \\ smths^{*} smooted_{x1_{min}} + (1 - smths) * x1_{min} & \text{otherwise} \end{cases}$$

The same is done for $y1_{min}, x2_{max}, y2_{max}$.
Utilizing these steps allows for videos where a team is always on display (4.3), with some videos still predicting spectators (4.4) and a newly introduced problem, namely skippers running out of view.
One way to solve jumpers running out of view, is by adding a default movement corrector, like the smooth value, which follows the new coordinates every frame. The movement corrector is based on the *iou*-value powered to the 1/8. It makes the crop following predictions by default, when IoU is getting low.

$SQRT = 8$

$movementCorrector = iou^{1/SQRT}$

[1]Smoothed values are obtained by playing around with values, choosing those leading to the best, subjective, smoothed review

$$smooted_{x1_{min}} = smooted_{x1_{min}} * movementCorrector + (1 - movementCorrector) * x1_{min}$$

A possible improvement could be matching individual boxes with the new predictions, eliminating the spectator. Which raises concerns about jumpers entering the competition field. There are no earlier predicted boxes to match. The total algorithm isn't perfect yet, but the results indicate sufficient valid video crops to use and experiment on segmentation and recognition C.5.
The results are broken down in table D.1. In this table you can see that about 15% of the videos aren't localized properly, some of them only have minor disturbs. This results in enough videos available to try and recognize skills.

4.2. Action segmentation

Experiments on action segmentation have waited on skill recognition (section 4.3), this is why the experiments with action segmentation use the same models as the skill recognition, adapting only the top layers of the models.

4.2.1. Video Vision Transformer & Self-attention ConvLSTM

The first and second experiment for segmention use code implementations of the Video Vision Transformer as introduced in Arnab et al. (2021) (ViViT - C.12) and the Self-Attention ConvLSTM from Z. Lin et al. (2020). To resolve bugs and dimensionality issues, Large Language models are used ((OpenAI, 2025) and (Deepseek, 2025)). These models were then trained on 7.8k batches of 16 consecutive frames, which originate from 44 trainings videos for about 48 minutes of training data, having frame rates between 25 and 60 [2] images per second. Both models showed the same results, averaging out predictions slightly above 0 for all frames, stagnating on MSE losses of around 0.09 and 0.10. This was about the average when always predicting 0. (Exact values are lost due to the switch to PyTorch and bugs, will/could be re-added later.)

4.2.2. Pre-trained MViT

In order to get better results on the little labeled available data, (44 videos for training, 7 validation videos), looking for a pre-trained model really helped a lot. PyTorch provides a list of pre-built and pre-trained models such as mvit-v1-b. Fine-tuning this model lowered the MSE to 0.065 with peaks overlapping splitpoints as demonstrated and elaborated in figure 4.6.
To get all sections, all peaks in its neighborhood ($windowsize = fps/3$) having a splitpoint value more than 0.4 become the start and end frame of a skill section, which will be used to recognize which skill is displayed. The code filtering segments out of raw-predicted splitpoint values is C.14, filtering peaks (fig 4.6) in a list of start and end frames. An example could be $[...2280, 2320, 2355, 2382...]$, as splitpoints.

[2]Most labeled videos have a frame rate of 25, 30 or 50 frames per second

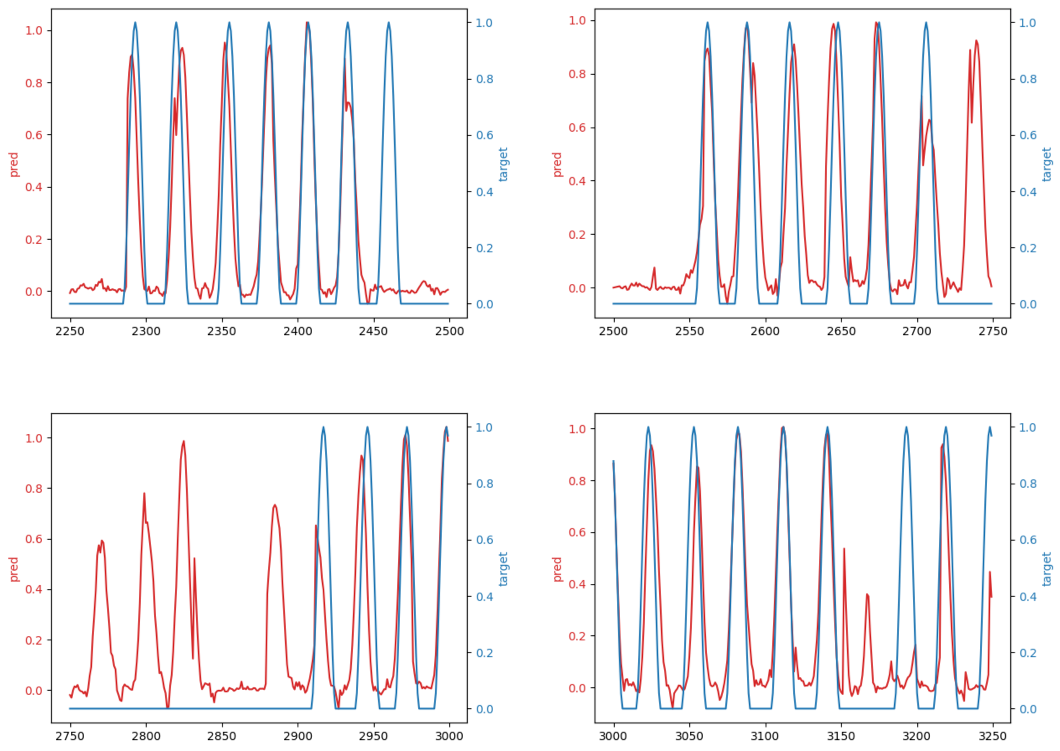


Figure 4.6: Segmentation plot comparing predictions with the splitpoint values indicating a match of interesting splitpoint. Visible errors are:
at frame 2459, a relatively lower jump out of the rope with a snapper.
at frames 2700 until 2900 indicating a transition which is labeled in one segment, but the athletes are rather bouncy on their feet, letting the model think those are splitpoints.
a handspring occurs from frame 3140 until 3192, but the model indicates too much during the skill, while it doesn't indicate the moment right after the handspring.

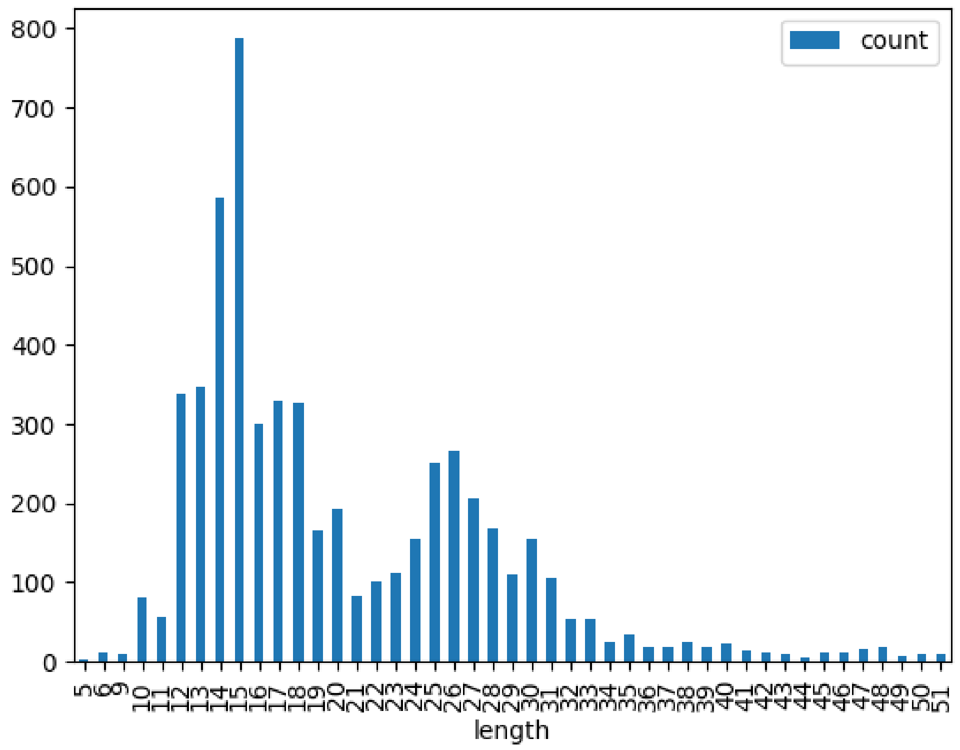


Figure 4.7: Distribution of skill length (in frames), visually limited to 50 frames per skill. Skill lengths differ depending on the frame rate of the video. Most videos have a frame rate of 25, 30 or 50 frames per second. Annotated sections longer than 50 frames are mostly transitions, mistake recoveries, gymnastic typed skills or power typed skills with more than one rotation. These long duration frame counts occur at most 3 times.

These predicted segments are the first usable split moments to use for splitting unseen videos recognizing or speeding up further labeling. This brings us to the main results of this paper, namely skill recognition results.

4.3. Skill recognition

Having small little sections of potential skills (normal jumps, and mistake recoveries included) allows to predict what actions are actually in this section. As the segments provided by the segmentation model have a varying frame count, and the length of skills in general can differ, they need to be transformed in equal sized frames. Depending on the frame rate, most skills have a frame rate between 12 and 18 frames per skill, see figure 4.7 for a distribution. In order to transform skills into equally sized sections, frames are skipped or duplicated depending on the length of the skill. The selected frames are calculated in code snippet 4.1.

```python
timesteps = 16
slot = skill_length_over_timesteps = skill_length / timesteps
frames [frames_skill[round(i * slot)] for i in range(timesteps)]
```

Listing 4.1: Code for frame selection of skills

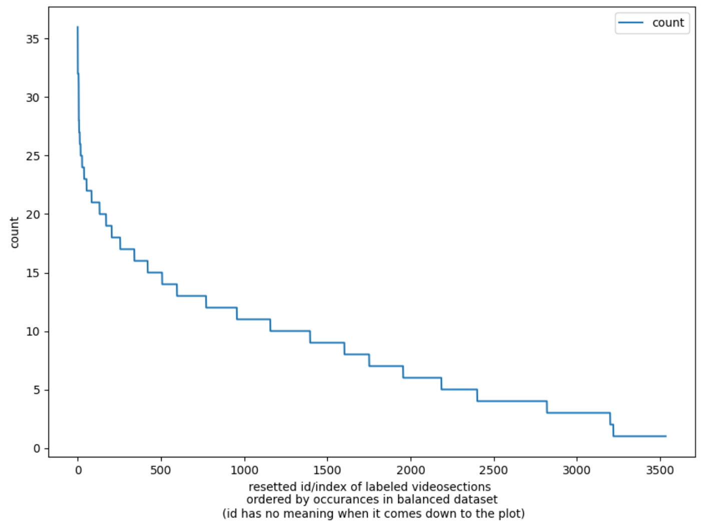
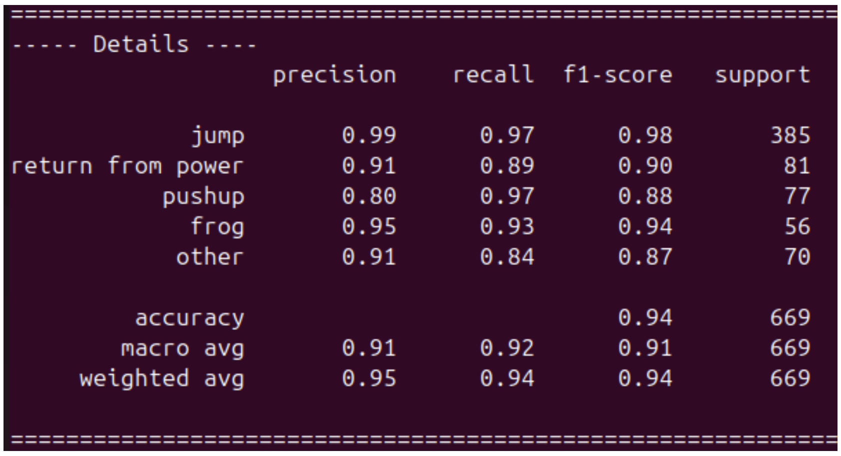


Figure 4.8: Occurrence of each labeled skill (e.g. videoId 123, frameStart 480, frameEnd 504, Double Dutch, 1 rotation, split, EB, crouger...) in the upsampeled dataset. Values get duplicated based on the uniqueness of each property. E.g. because it is an EB and a split, it will be duplicated four times, but no/less duplication because Double Dutch and 1 rotation is more common.

4.3.1. Video Vision Transformer & Self-attention ConvLSTM

Using about 5k skills, 4550 to train and 669 to validate, the same video models as described in the segmentation phase 4.2.1 can be trained the skill properties defined in 2.3. During the annotation of videos, it is noticed how screwed the skill-data is. Other action properties like turners, hands, type or faults are equally skewed. See table 4.1 for a breakdown of skill occurrences.
Both the ViViT and the SA-ConvLSTM were plateauing on the most appearing value, always predicting normal jumps, normally turned, without a mistake etc. Even upsampling less occurring combinations didn't help. The models stagnated on validation losses of 5.9 and 6.5. See figure 4.8 to see how many times a skill appears in a training round, depending on its uniqueness.
In order to reach better results, the search for a pre-trained model leads us to the following experiment, namely the usage of PyTorch implementation of the Multiscale Video Vision Transformer.



```
==========================================================
----- Details ----
                   precision    recall  f1-score   support

              jump      0.99      0.97      0.98       385
return from power      0.91      0.89      0.90        81
            pushup      0.80      0.97      0.88        77
              frog      0.95      0.93      0.94        56
             other      0.91      0.84      0.87        70

          accuracy                          0.94       669
         macro avg      0.91      0.92      0.91       669
      weighted avg      0.95      0.94      0.94       669

==========================================================
```

Figure 4.9: Classification report on the 5 classes jumps, pushups, return from powers, frogs or other skills after training MViT on an equally distributed, max 442 samples per skill as seen in table 4.2.

4.3.2. Multiscale Video Vision Transformer - MViT

The implementation of the the Mutliscale vision transformer uses the PyTorch pre-trained model and weights C.17. The output layers of the model are stripped in order to adapt the model to recognizing jump rope skills. This allows the addition of new output layers, custom to DD3 freestyles C.16.
PyTorch's Video MViT is an implementation based on the Multiscale Vision Transformer created by Fan et al. (2021) which can be fine-tuned using the pre-trained weights on the kinetics dataset. The first experiment using this model used a perfectly balanced dataset for skills only, namely taking a random subset of maximum of 442 samples on each of the 5 classes seen in table 4.2. The first results showed accuracies of 87% to 98% (see fig. 4.9) even some turner skills getting predicted correctly.
Re-enabling all skill classes and all other properties results in a total validation loss of 1.3835, as compared to 5.9 or 6.5 earlier. For skills only, the weighted f1-accuracy is 93%, while macro f1-accuracy is 40%. The macro accuracy gives a better indication as accuracies on all defined classes, e.g. pushup, jump, swift, are taken into account equally. In this experiment, the less appearing skills lower the score of the macro accuracy, as they do not appear as often.
Like the previous subsection, these results are trained on the skill distribution displayed in table 4.1). A breakdown of all classification reports can be found in tables D.2 -> D.14. You may notice that the predictions for bodyRotations, sloppy, hard2see & faults have yet to start learning.

4.3.3. Skill recognition - Act as a judge

Being able to recognize skills allows to assign respective levels on combinations. After assigning the proper levels, the respective scores can be assigned mapping based on the total levels jumped. Table 4.3 illustrates how many points each level

Train %	Train Count	Skill	Val Count	Val %
50.6687	2273	jump	379	56.6517
14.0660	631	pushup	77	11.5097
13.0629	586	return from power	84	12.5561
9.8529	442	frog	59	8.8191
3.2100	144	crab	9	1.3453
2.0062	90	split	12	1.7937
1.1146	50	flip	5	0.7474
0.9140	41	rondat	5	0.7474
0.8917	40	rad	3	0.4484
0.8025	36	suicide	8	1.1958
0.7356	33	handspring	5	0.7474
0.6910	31	rol2kip	7	1.0463
0.4458	20	kip	6	0.8969
0.3567	16	speed	NULL	NULL
0.3121	14	kopkip	1	0.1495
0.2675	12	roll	3	0.4484
0.1783	8	stut	2	0.2990
0.1337	6	swift	NULL	NULL
0.1337	6	UNKOWN	1	0.1495
0.0669	3	leapfrog	NULL	NULL
0.0446	2	footwork-kick	NULL	NULL
0.0223	1	mountainclimber	NULL	NULL
0.0223	1	footwork-open	1	0.1495

Table 4.1: Skill distribution with training and validation counts and percentages. Null values indicate absence of skill occurrence in the validation data.

receives.
The first results are shown using recordings of national finals. With an average score difference of -32.17%. The first score mappings don't look that good. Inspecting these results further, some videos indicate more than -70% score difference. Inspecting these results show invalid crops predicting spectators, potentially because of a different sport gymnasium and a lower camera perspective, compared to other videos. Another competition will be put next to these results, in order to gain more insights.

Skill	Train Count	Train %	Val Count	Val %
jump	2273	50.6687	379	56.6517
pushup	631	14.0660	77	11.5097
return from power	586	13.0629	84	12.5561
frog	442	9.8529	59	8.8191
other	526	11.7253	68	10.1645

Table 4.2: Train and validation skill distribution with low-frequency skills grouped as ”other”

Level	Score
0	0
1	1.5
2	2.2
3	3.3
4	4.9
5	7.3
6	11
7	11
8	11

Table 4.3: Corresponding score for each skill level, in Belgium, levels are limited to 6, any higher level remains a level 6. (Applicable during seasons 2023-2024 and 2024-2025)

4.3.4. Weighted loss

In order to increase accuracy on lower represented classes. To enforce this, losses wil be increased for mistakes on lower represented classes. A little mathematical magic C.20 based on the value counts of each skill aspect compared to highest occurance, applying some buffer, taking the difference, squaring, comparing to the mean etc. (C.20) allows for weight distributions below.

- loss weights for Type [1.0000, 1.5151, 4.7846, 4.7938, 4.2562, 4.7383, 4.6330]
- loss weights for Rotations [4.4372, 1.4893, 3.9782, 4.4880, 4.5276, 4.5960, 4.5960, 4.0000, 4.5997]
- loss weights for Turner1 [1.0000, 1.4437, 3.8602, 3.6128, 4.1699, 4.1746, 4.1079, 4.1067, 4.1866, 4.1711, 4.1818, 4.1818, 4.1890, 4.1902, 4.1878, 4.1926, 4.0000, 4.1914, 4.1914, 4.1938, 4.0000, 4.1926, 4.1902, 4.1938, 4.1938, 4.1938, 4.1938]
- loss weights for Turner2 [1.0000, 1.4450, 3.8687, 3.6231, 4.1794, 4.1890, 4.1026, 4.1325, 4.1938, 4.1866, 4.1854, 4.1926, 4.1926, 4.1986, 4.1950, 4.2023, 4.2023, 4.1974,

videoId	judges	MViT	Differences
2568	169.26	147	-13.15
2569	181.23	166.9	-7.91
2570	146.71	25.4	-82.69
2571	180.58	170.8	-5.42
2572	251.14	68.7	-72.64
2573	141.37	9.7	-93.14
2574	272.28	227.4	-16.48
2575	331.22	257.9	-22.14
2576	210.86	207.3	-1.69
2577	261.63	130.9	-49.97
2578	128.6	135.6	+5.44
2579	203.48	196.4	-3.48
2581	216.81	89.6	-58.67
2582	259.88	223.7	-13.92
2583	348.27	168.4	-51.65
2584	366.94	177	-51.76
2585	68.31	54.6	-20.07
2586	175.59	159.9	-8.94
2587	260.29	192.1	-26.2
2588	319.09	231.4	-27.48
2589	148.5	108	-27.27
Total	4642.04	3148.7	-32.17

Table 4.4: Score assigned by judges compared to the prediction the MViT model
Videos are on fully unlabeled data and from the same competition day, with a camera perspective quite a bit lower than other times. This results in quite some videos with improper crops.

4.1986, 4.0000, 4.2011, 4.2011, 4.1998, 4.0000, 4.2035, 4.0000, 4.1998]

- loss weights for Skill [1.0000, 1.4673, 2.5125, 2.5218, 2.5897, 4.0175, 4.4216, 3.8261, 4.4017, 4.4216, 4.2631, 4.2532, 4.2697, 4.3224, 4.3752, 4.2236, 4.4216, 4.2828, 4.3719, 4.4150, 4.2598, 4.4216, 4.4183, 4.3786, 4.4183, 4.3719, 4.4183, 4.0000, 4.0000, 4.0000, 4.3984, 4.4216, 4.4216]
- loss weights for Hands [2.0090, 7.7774, 3.2177]
- loss weights for Feet [5.6549, 5.2853, 1.7504]
- loss weights for Turntable [1.6041, 5.5475, 5.6556]

- loss weights for BodyRotations [1.5966, 5.6104, 5.6066]
- loss weights for Backwards [1.7598, 7.1472]
- loss weights for Sloppy [1.7704, 7.1258]
- loss weights for Hard2see [1.7622, 7.1423]
- loss weights for Fault [1.7756, 7.1154]

These weights are then given to CrossEntropyLoss of PyTorch for the categorical outputs and given to a customized MSE C.21, for numerical and boolean outputs, incorporating these weights. This resulted for Faults, Sloppy and Hard2See to start being recognized D.15 and pushed the total macro f1 avg accuracy to 51.53%, which is about 6% higher then before.

4.4. Video ResNet and Swin Transformer

To top it all, additional models where added to fully compare the performance of the MViT model. The first additions are the PyTorch implementation of video ResNet models, based on the paper of Tran et al. (2017). All three of the adaptions, R3D, MC3 and R2plus1 were stripped from their head and filled with the jump rope skill output layers, just like the MViT implementation. Another model listed by PyTorch was the Swin Transformer, created by Liu, Ning et al. (2021), implemented the same way. Only the tiny (t) and small (s) versions of the Swin transformer have been tested.
Utilizing the pre-trained weights on kinetics 400, only the Swin Transformer reached similar results and even bested the Multiscale Vision Transformer by a few percentiles (52.36%), which you can see in table below 4.5. However, the MViT models stays strong, keeping the highest total accuracy (93.83).
Using these increased accuracies on recognition & other models, judge scores can be compared along the models. To keep things uncluttered, only the total deviation is showed now. While the swin transformers showed more potential in the macro recognition of skills, the multiscale video transformer outperformed on similarity with the judges.
You may have noticed, that the final run decreased the score difference from -32.17% to a whopping -21.68%, finalizing the results in a positive note, being able to discuss and summarize this research.

model	macro	m skills	weighted	w skills	total accuracy
Resnet_R2plus1	25.64	2.70	70.3	25.51	71.87
Resnet_MC3	25.98	1.39	69.49	9.85	69.88
Resnet_R3D	27.43	3.52	70.24	19.76	71.22
SwinT_t	49.13	32.06	91.28	89.4	91.99
MViT	51.53	27.75	93.11	90.7	93.83
MViT_extra_dense	51.96	31.46	93.1	91.45	93.73
SwinT_s	52.36	36.73	92.8	91.79	93.3

Table 4.5: macro = f1 avg macro accuracy
m skills = f1 avg skill accuracy
weighted = f1 weighted avg
w skills = f1 weighted avg skills
total accuracy = true positives / total aspects

MViT	MViT dense	Swin s	Swin t	MC3	R3D	R2plus1
-21.68	-20.94	-23.83	-27.72	-38.85	-53.09	+47.15

Table 4.6: Score assigned by judges compared to the prediction vision models
The multiscale vision transformer is clearly winning on assigning scores.

5 Discussion

The goal of this research was to increase in the objectivity and accuracy of scores assigned by judges on jump rope freestyles during competitions using recent advancements in machine learning technology. In order to achieve this, two sets of questions needed answers. The first set being questions about jump rope, which will be answered in 5.1. The second set contains questions about the machine learning part, which follows in 5.2. Some of them required an answer before starting the development of the proof of concept, while others depended on the PoC.

5.1. Jump rope answers

Today, the challenge for jump rope judges 1.2 is the ability to keep up with the skills performered, while calculating the skill level, in order to accurately score the difficulty of a routine. In order to reduce errors, judging methods change, such as adapting the rules 2.1.3 or reviewing the routine post-performence at slower speeds 2.1.4.
Skills were then broken down 2.2 to better understand how difficulty is scored 1.2. These were further specified and organized into a skill matrix 2.3 in order to specify which exact actions needed to be recognized 1.2. Having skills specified, leads to the machine learning part, transforming this matrix into computer output and exploring machine learning possibilities.

5.2. Machine learning answers

Computer vision 2.4, a subset of machine learning, is the ability of computers to understand visual data. Exploring this topic; models, other sports, the jury support system of Fujitsu (1.2, 2.4.1), the availability of video recordings (1.2, 2.3), along with acquiring information about and NextJumps' speed-counter 2.4.2 allowed for the choice to learn out of video material 1.2.

Using video data and recognizing actions required to think about quite some properties. Recordings are typically full routines of about 60 to 75 seconds. These freestyles contain 40 to 60 skills, around 100 if you include normal jumps. Some recordings are zoomed-in, others are stationary capturing the whole field. The video type may be mp4, blu-ray (m2ts) or AVI. Even the frame rate could differ (25, 30, 50). Considering these aspects, a general approach for recognizing skills in a video is created while exploring human action recognition 2.5. This resulted in three steps; jumper localization 2.6, action segmentation 2.7 and skill recognition 2.8, each of them requiring future work 5.3.

5.2.1. Localization

Localization serves as a way to zoom in and focusing on the actions, sparing computational resources, rather than providing frames to a model where the actions and athletes only take up a limitted amount of space in the video 3.1.
on the Tests on results for localizing 4.1 are limited in order to have spent sufficient time for segmenting and recognizing skills. Even then, time was limited. Predicting the location of jumpers has been tested using full boxes as a first stage. Models in this stage included MobileNet and GoogleNet, but results were lacking. The switch to annotate individuals instead of full teams and using a pre-trained YOLO model quickly reached better results. Even though the mAP50 reached 0.943 accuracy, initial full videocrops weren't stable, requiring smoothing techniques and more labels in order to reduce spectator predictions and visually disturbing shocks (table D.1). Utilizing a basic smoothing technique 4.1.2 allowed for sufficient videocrops, which can be used for skill recognition & segmentation.

5.2.2. Future work on localization

Localization can be improved in multiple areas. This could involve training on full team labels using YOLO, applying other smoothing techniques or implementing other models. Other factors could be changing the competition setting to limit the amount of spectators, keeping the camera perspective consistent or provide more different perspectives. Another way is annotating the predicted spectators or judges, indicated by pre-trained models, which can be filtered out when cropping or properly managing the location of recording cameras on competition settings.

5.2.3. Segmentation

The part about action segmentation definitely needs more research and experiments. The easiest implementation was utilizing vision models used for recognizing actions to annotate interesting split points having a value of 1, a split moment, or 0, executing a skill or doing nothing. Using the Multiscale Vision Transformer, developed by (Fan et al., 2021), which acquired great results recognition 4.3, quickly showed overlap between predicted split points and labeled split points 4.6. The

MViT applied for segmentation transforms the annotated skill labels 3.4 into split labels 3.3. A little summary of the labels follows. Skills labels have a start and end frame, which allows to indicate split values at these frame numbers to be around one. An example would be:
$[0, 0, 0.1, 0.31, 0.55, 0.86, 1, 0.86, 0.55, 0.31, 0.1, 0, 0, 0 \dots]$
Videos were then split into partitions of T = 16 timesteps, learning some context of previous and following frames, in order to predict the corresponding T split output values as well. The highest peaks were then considered split points 4.6. Being able to predict splitpoints allows to perform skill recognition on completely unseen videos, isolating each action.

5.2.4. Future work on action segmentation

Due to time constraints and priorities, only the MSE has been used to decide the most optimal split moments on the MViT model. There are still future actions which could improve action segmentation. Ideas, not limited to the list below, include:

- Implement other metrics for segmentation next to MSE
- Wider or more narrow split point values.
  - IoU section overlap
  - Average distance to closest split point
  - Count of predicted splitpoints vs actual splitpoints
- Implement a second dimension, is jumping parameter to filter out mistake recoveries or non jumping moments.
- Predict every other frame, instead of all frames.
- Implementing a action segmentation specific models.
- More labels

5.2.5. Recognition

Being able to zoom in on athletes and having a model which can segmenting actions, actual skills can be predicted on full videos. But even before the ability to isolate skills, labeling and training experiments on skill recognition are possible. One minor issue, solved in the methodology of skill recognition 3.4 was the fact that performed actions have a different length 4.7. This is solved by ensuring they have the same length of T = 16 timesteps, skipping or duplicating frames 3.4. This resulted in equal inputs with a dimension of (channels, timesteps, height, width), (C, T, H, W) for short.
Next up was transforming all aspects of skills into machine understandable output labels. This was the reason why the skill matrix has been created 2.3. The matrix

resulted in a skill configuration C.8 for skill labels, specifying 13 different outputs; Type, Rotations, Turner1, Turner2, Skill, Hands, Feet, Turntable, BodyRotations, Backwards, Sloppy, Hard2see and Fault. Each aspect being a category, numeric value or a boolean.
The final adaptation using weighted losses, which returned a different loss depending on the occurrence of skill aspects pushed the MViT model into grasping the meaning behind the fault, sloppy and hard2see aspects. It raised the f1 average macro accuracy from 45.61% to 51.53%, giving in on f1 macro skill accuracy. It fell from 34.93% to 27.75%, which is low compared to the skill recognition accuracies on MViT extra dense 31.46%, SwinT t 32.06%, and SwinT s 36.73%. However, it still reached highest on the normal accuracy measure 93.83%, compared to 93.73%, 91.99%, or 93.3%.
It is unknown how much data is expected to increase the accuracy 1.3, but one major improvement is adding sufficient examples of each skill or turner, in order to provide enough feedback to model to learn more about these less represented skill aspects.

5.2.6. Future work on skill recognition

Aside from additional labels, some possibilities to increase accuracy are fine-tuning the weighted losses or experimenting using different models. Another experiment could be predicting the actions of individual athletes, instead of regarding the video section as a single unit. This can be especially usefull when predicting single rope team freestyles, in which the jumpers act as a unit, but also act as an individual, contrary to double dutch freestyles. A model like stagNet, (Qi et al., 2020), looks interesting.
Currently the data has been downsampled, filtering mostly normal jumps. This downsampling can be adjusted, as it there could be turner skills when there is a normal jump, limiting the training of these turners. Further data augmentation, besides mirroring the visual, such as color shifts, random crops or camera motion can also increase the accuracy of the models. A more difficult research topic, could be to create a full 3D pose of the athletes, which might require multiple cameras.

5.2.7. Judge score

In the final run, MViT reached the best judge score results, with a difference of -21.68% compared to the scores allocated by judges. Time constraints didn't allow for a comparison of judge scores and model predictions with the ground truth. Thus the desired outcome to have a score difference around 5 to 10% couldn't be measured. Even the -21.68% is way to far off in order to use at competitions and unacceptable on competitions. Remember that the scores of scores are only a proxy for the actual score of a routine.

5.3. Other future work

When ground truths are available for the routines, an actual comparison between judge scores and models can be made to decide the better judge. Meanwhile, skill matrices for other events can be created, labeled on videos and trained, allowing for a broader range of models, model outputs and possibilities.
To conclude, this research showed the possibilities of the proposed three step architecture for recognizing skills in a full routine on a limited dataset. While results aren't perfect, great effort has been put in the proof of concept, enabling future research. The main priority should now be recording and annotating more videos, with different camera perspectives, including more skill of each skill variation in both the training and validation dataset.

A

Research proposal

The subject of this applied thesis is based on the earlier approved research proposal. This submission is added as attachment.

Proposal abstract

Judging jump rope freestyle routines at the highest competitive level has become increasingly challenging due to the evolution of jump rope. Both the number of skills that are included in a routine as well as the speed with which these are executed keep increasing. This is particularly evident in so-called Double Dutch Freestyle routines, which is why assigning scores to these freestyles is done by a combination of live and delayed evaluation. The creativity of a routine (including its variation and musicality) is scored in real time but the assignment of the appropriate difficulty level is done based on a recording of the routine replayed at half speed right after it is performed. Even though this helps reduce errors in difficulty scoring, a certain variability in the assigned scores persists/can still be seen. With the increased accessibility of artificial intelligence, particularly neural networks, the question arises whether an AI judge or assistant can be developed to obtain a more accurate (objective) difficulty scoring.
This research explores the possibility and development of such an AI assistant, as well as the techniques and challenges required to obtain the desired level of objectivity. The current idea is divided into three sections. The first section will be localizing the jumpers in the field as most obtained recordings are not fully zoomed in or recorded using a static camera. As recorded jumpers sometimes take up less than a fifth of the recording, they can be cropped out sparing computational resources for the parts to come. The second part involves isolating skills from a routine into individual skills or subskills. This enables the assistant to not only label a single skill, but also dozens of skills performed sequentially without interference. Lastly, each

segment can be assigned to its corresponding skill. For Double Dutch Freestyles this means the combined action of jumpers and turners resulting in a large possibility of unique combinations. By further marking presentational skills or difficult to see skills as unknown (e.g. when one athlete stands between others and the camera) it is expected that the AI judge will indicate unknown or unclear skills by itself. This way the the assistant can be put into practice when reaching similar or more accurate results than the live jury panel. The assistant's results allow for verification of scores during and after the competition, increasing transparency and accuracy even further. Using the current obtained competition videos displaying the most common skills a few hundred times, it is expected that the AI judge will start to distinguish between skills like a cartwheel, split or salto. In case it works, it is not only useful for jump rope freestyles but also applicable in other judge-related competitions such as gymnastic routines, figure skating, or synchronized swimming.

A.1. Introduction

Jump rope is an evolving sport. Year after year, an increasing amount of high-level competitors are pushing the limits of jump rope. This results in new skills, new combinations, better physiques, better rope material, and faster movements. For the judges to keep up with the jumpers and to correctly assess scores to a routine, Double Dutch freestyles[1] are reviewed at half speed in International competitions or even at nationals in Belgium.
Head judges around the world question the best way to judge athletes correctly so as to give an accurate and objective ranking in national or international competitions. Many solutions have been provided: another judging rule set[2], splitting judge responsibility, replaying the routine at half speed ...
However, with the increasing popularity of image recognition, more powerful computers, applications that recognize objects in images (Singh Gill et al., 2022), implementations detecting simple human actions (Luqman & Elalfy, 2022), examples of action recognition in other sports (Yin et al., 2024) and the successful test of NextJump's AI speed-counter, head jurors wondered about the possibility of using modern technologies like artificial intelligence to improve the judging results of jump rope freestyles.
This results some in the main research question of this paper: **"How can artificial intelligence be incorporated into jump rope freestyles to increase the objectiveness and accuracy of judging?"**.

[1]Two turners, with one or more jumpers
[2]The current rule set is enforced and maintained by Gymfed, closely related to the international judging-rules from the International Jump Rope Union

A.1.1. Problem domain

Jump rope, like gymnastics or athletics, exists in many disciplines like Speed, Single Rope freestyle (single/pair/team), Chinese Wheel (CW) or Double Dutch (single/pair). To accurately judge routines, each discipline has its own rules. Although interlapped, differences exist. Each athlete or team then performs a choreography of 60 to 75 seconds showcasing their favorite skills and uniqueness.

What are the challenges for judges today?

On competitions, judges watch the routine live to annotate the difficulty or creativity. Each judge then pays attention to his assigned part, e.g. movement, musicality or difficulty, all of those elements contributing to the total score of the freestyle. The main problem now is for judges to keep up with double dutch routines. Single rope freestyles are still manageable. To increase the accuracy, difficulty-certified judges [3] are already allowed to review freestyles at half speed in order to score them accurately.

How is difficulty scored?

Performed skills each have a level, which will be written down when seen by the judge. Each level also has a numeric score that contributes to the total difficulty of the routine. Judges see and calculate/memorize the level of each skill, write it down, count the number of levels jumped and calculate the diff score.

What are the skills and transitions that need to be recognized?

For double dutch, there are two turners and at least one jumper. All three of them act as a unit and execute skills or turner combinations. Jumpers can do a handstand, push-up or a cartwheel, while turners can turn with their arms crossed, on the back or rotating two ropes underneath the jumper at once. Furthermore, combining skills of a jumper while doing a special rotation contributes to more difficulty and more points. Even different transitions like a turntable, which is a push-up to a push-up while your body position changes a quarter is a different transition.

A.1.2. Solution domain

Until now, only the scope has been narrowed down. The focus will be put on double dutch freestyles. Getting to know the topic is great, but this doesn't get us further into the solution. Let's jump into it.

Which modern technologies can be used to increase score accuracy of jump rope freestyles

As reviewing routines at half speed still has his limits, using machine learning (an AI-model) could reduce time spent on judging routines. The idea would be a machine learning model recognizing (sub)skills and transitions in a double dutch single freestyle. (DD3)

[3] Those judges judging the difficulty of double dutch routines.

Which data is available for the machine learning model to use?

To recognize hundreds of skills, variations and transitions, lots of data is needed. Both individual and team freestyles are mostly recorded by clubs themselves or event organizers, some which are available on social media. The task is to explore and gather as much as possible.

When are predictions acceptable to potentially use on competitions?

Judges do make mistakes, just like the machine learning models, but we do need a baseline for acceptable results. Past competition scores can be used to define a target.

How can the AI-judge as a hybrid model increasing judge quality of the judges?

Can the AI-judge [4] be used to train new judges or brush up the knowledge of current judges? Meanwhile, can they verify the predicted labels by the model to use as new training data?

Which activity recognition examples can be used or altered as a base guidance?

Quick searches give examples of object recognition (Diwakar & Raj, 2022), detecting sign language (Bora et al., 2023) or activity recognition (e.g. on the kinetics dataset - riding a bike, reading a book, playing an instrument (Kay et al., 2017)). These implementations can be used as a first guide. Those examples give the idea that data mostly seem more to be centered, which is not the case in jump rope videos, a solution needs to be found for that. The second problem is that freestyles can consist of more than fifty different skills, which takes a long time to cut manually.

What would be a minimal Proof of Concept (PoC)?

The PoC would be a model recognizing the most common skills and transitions. This could mean omitting or just marking special combinations, longer double dutch switches or long time sequences of emptiness in general. Preferably, the PoC should be able to generalize uncommon skills that are still definable as normal. Better described would be knocking on the door three times, someone's at the door, but knocking four times or with a bonk in between is also recognizable [5] for a more concrete case.

How much data is expected to increase the accuracy off the Judge?

The amount of videos will keep rising, but will the current amount be sufficient? If it's not enough, how much more would be expected and what about uncommon skills. Do we need to specially record them? But what about new skills on competitions?

[4]The machine learning model predicting performed skills
[5]See specified example in section A.2.10

A.1.3. Additional questions

The proof of concept will probably raise a lot of questions as a byproduct such as:

- How can we use the AI-Judge to improve judges?
- What needs to change on a working model, to apply it on other judge-related sports such as gymnastics, synchronized swimming, figure skating ...

A.1.4. Introduction summary

With some general knowledge about jump rope and thethesis goal defined, further research, (label)definitions, model selection, and implementation can be performed. Let's start by exploring earlier work while slowly increasing the number of jump rope definitions.

A.2. Literature review

The research towards skill-recognition will be done by steps. First will be an introduction of skills, after which computer vision will be explored along with NextJump's speed-counter. With a general proposed flow in mind, enables an improved research towards specific models and fine-tuning of the expected approach and potential challenges to recognize skills.

A.2.1. Skills intro

Earlier we described the presence of multiple disciplines in Jump Rope. To keep the research doable, skill recognition will be started for one discipline, namely DD3 freestyles, which already contains some Chinese wheel integration.

Double Dutch Single Freestyle - DD3

DD3 consists of two turners and one jumper alternating ropes. Elements in Double Dutch are similar to single rope, but different at the same time. The jumper does all the skills, mainly powers, gymnastics or footwork, since he doesn't need to hold the rope. Turners can manipulate the rope using multiple unders, turner-skills like crossed arms, EB, toad...or even involve gymnastics themselves. To judge double dutch, 'snapshots' are taken, then the corresponding level will be given depending on the combination of turners, skills and rope-rotations.
Like any discipline, mistakes can happen, they'll be deducted from the total score. Some examples with their current corresponding levels in double dutch.

- Powers
  - push-up - to plank position and pushing upwards while pulling the rope underneath your feet. (lvl 2)
  - split (2)

  - frog - handstand (2)
  - swift/V-kick (3)
- Gymnastics
  - cartwheel (2)
  - kip (3)
  - salto (4)
- Turners
  - cross (c - crossed arms on the stomach) (+2/+0)
  - crouger (raise knee, put your arm underneath it) (+1)
  - EB (arm on stomach + arm on back) (+1)
  - TS (arms crossed behind the back) (+1/+1)
- Multiples
  - double - DU - 2 rotations (+1)
  - triple - TU - 3 rotations (+2)
  - quad - QU - 4 rotations (+2)
  - quint - 5 rotations (+3)

Each power can then be varied one handed, as a turntable, as a consecutive ..., which gains extra levels. Judges calculate or memorize the whole level of each skill/transition. Without further context, annotations like (+1/+1) can already seem confusing, well it is.

A.2.2. Challenges of judging

As introduced earlier, based on own experiences and statements of colleagues, the sport is evolving. These statements are supported by commentary from the IJRU world championship livestream day 1 (IJRU, 2023a) to day 8 (IJRU, 2023b). Speed records are slowly rising, see table A.1 or A.2 [6], also quads or quints in single rope freestyles are becoming the norm, where 10 to 15 years ago, it was considered a wow factor. This is also the case for double dutch; more variations, more turner involvements, faster and longer skill-sequences etc. All which need to be perceived using the same old brain capacity of a judge. To help judges and improve, some actions already took place.

[6](Laue, 1999), (Gymfed, 2017), (IJRU, 2024b), (IJRU, 2024a)

Year	World	Europe	Belgium	Usa	Hungary	Germany	China
1998							
1999	80		80				
2012							
2015							
2016	111	103				103	
2019	111		102	105.5			
2020	111		102	105.5			
2021	111		103.5	105.5			
2022	111		103.5	105.5			
2023	113		103.5	106			113
2024	113	108	103.5	106			113

Table A.1: History of speed records males

Year	World	Europe	Belgium	Usa	Hungary	Germany	China
1998	83				83		
1999							
2012			102				
2015	105	105			105		
2016	105	105	102		105		
2019	108.5	105	102	100.5	105		108.5
2020	108.5	105	102	100.5	105		108.5
2021	108.5	105	102	100.5	105		108.5
2022	108.5	105	102	100.5	105		108.5
2023	108.5	105	102	100.5	105		108.5
2024	108.5	105	102	100.5	105		108.5

Table A.2: History of speed records females

Splitting responsibility

With the current rules, judges are devided in two main categories, those judging difficulty and those judging creativity. Creativity is further split into execution, entertainment, musicality and variation. This breakdown allows increased attention on different aspects of a routine, thus decreasing potential observation mistakes.

Multiple panels

Using two ormore judge-panels, more freestyles can be evaluated at the same time. While one panel is watching a freestyle, the other can summarize and calculate the total score. Having the same panel judge the same category, e.g. juniors vs seniors, also decreases the effect of differences as a people between judges, e.g. being more strict, less observational moment(s), incorrectly memorized a skill-level etc.

Adapting the rules

Changing the rules about how a freestyle must be evaluate, can impact the way of thinking, memorizing or calculating the level, score or deduction of a skill. This was tried by using the current 'snapshot' system for double dutch. This was still perceived as hard based on reactions of fellow exam takers in 2023, 2024.

Review at slower speed

In recent years, on world competitions or on some local competitions in Belgium, the video replay was introduced to review a double dutch freestyle at slower speed to accurately assign level performed. As this can be time consuming, another separation in the judge panel, extra diff judges, was created to give judges enough time to review the routine. Even in slow motion, it's still perceived to be hard to see all the actions of all the jumpers, while calculating the total level of the base skill, transition, turnerskill and rotation speed, if it's not a repetition[7].

Challenges summary

Incorporating all these things brought jump rope to where it is. To make our lives easier, we try to find improvements. One of these is exploring automatic skill-recognition by using AI. When skills are recognizable by a program, they can be mapped to a corresponding level contributing towards the end score. Knowing what's represented in an image or a video is called computer vision.

A.2.3. Computer vision

To automatically recognize skills, input data is needed. There are quite some videos on socials, as well as in-house recordings, see table A.3, so the choice to learn from videos was made quickly. This is also what motivated NextJump. Computer vision is the field of study in which computers recognize features, people or other objects in digital imagery. More specifically, the focus is recognizing human actions in these recordings, called human activity recognition or HAR (Pareek & Thakkar, 2020).

[7]Repeated skills don't contribute to the score

	SR	DD3
#Freestyles	500-1000+	286-352
Hours	8h-16h+	5-6h
Years	500 from 2024	mainly 2020-2024
MVP	basic variation elements	basic powers, gyms, turnerskills
Level-guessing	0 to 8	0 to 8
Theoretical level limit	8+ levels possible	8+ levels possible
Variation elements	6	4
skill-matrix	more complex	simpeler compared to SR
longer sequences	/	/
individuals	1	3
competitions	Oct-Nov	March-Apr

Table A.3: Data comparison

Other adaptations like Human Gait Recognition, HGR, or Human Pose Estimation, HPE, are also used to recognize human activities. Although the three techniques are closely related, each of them has a different nuance. Gait recognition looks at a person's typical movements, gestures or behavioral patterns (Alharthi et al., 2019), while pose estimation looks specifically at poses or special expressions (Song et al., 2021). They also talk about how pose recognition, e.g. skeleton-based, can be used as a tool are to improve activity labeling.

Computer vision in other sports

Soomro and Zamir (2014) published a book about the first advancements in computer vision since it was applied to sports. Lots of history, where topics like dimensionality reduction is more discussed and a topic then, compared to now. On the other hand, Yin et al. (2024) focuses themselves on the latest advancements of computer vision in sports, mainly focused on teams. Although relatively popular, neither of them really talks about gymnastics, which is closer related to jump rope in comparison with sports like tennis, basketball or cricket. Given examples by those two sources of performed computer vision tasks are player tracking, following the ball trajectory (e.g. in cricket, football, tennis), human to human interaction (e.g basket) or detecting action types (e.g. running, walking, hitting the ball) or predicting the sport itself. (e.g. Olympics dataset)
To near closer towards jump rope, Abdullah and Alsaif (2023) provided an example of static image recognition in which gymnast poses on the rings where predicted, although in a balanced dataset and limited amount of classes. More intensively, score prediction was performed by Zahan et al. (2023). This is already close

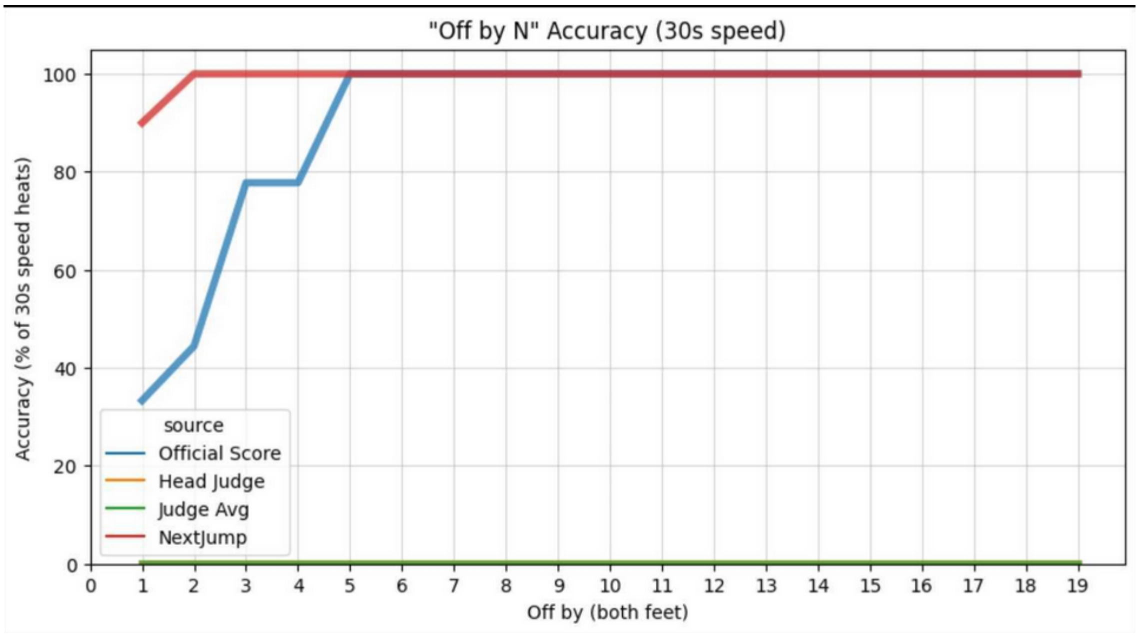


Figure A.1: Comparison of avg scores given to a jumper, compared to the effective score. Results are from 2024 AMJRF nationals.

to what is wanted, however, he modified the LSTM-model to incorporate longer time sequences, to predict a full score of a routine which is static and not future proof. Changes in rules would make older scores useless and request for mistakes. Combining these and earlier papers, detecting which action is performed, with a level/score mapping afterwards will be better future proof.

NextJump Speedcounter

As of august 2023, NextJump tested their AI-speed-counter on the world competition acquiring, really accurate results, see fig A.1 & A.2. They found that 10 hours was sufficient for a single event (e.g. just single rope speed) but to count all kinds of events more (diverse) data is needed[8]. Using this as a base/guidance, the likelihood to succeed implementing skill-recognition in freestyles grows. When skills are recognized, they can be mapped to their corresponding level and summed up to achieve the total score of a freestyle.

A.2.4. HAR general progress

Pareek and Thakkar (2020) did some research about the recent updates in human activity recognition, which mainly gave form to the following general approach. As jumpers can stand everywhere in a field[9], locating and cropping athletes can improve the segmentation model[10]. When the skippers are centered, action segmentation can be performed. This allows predicting skills on newly recorded videos,

[8]The current dataset entails 36h video material
[9]A field is generally 12x12 or 15x15 meters
[10]If time allows it, the final model can be compared with or without localization

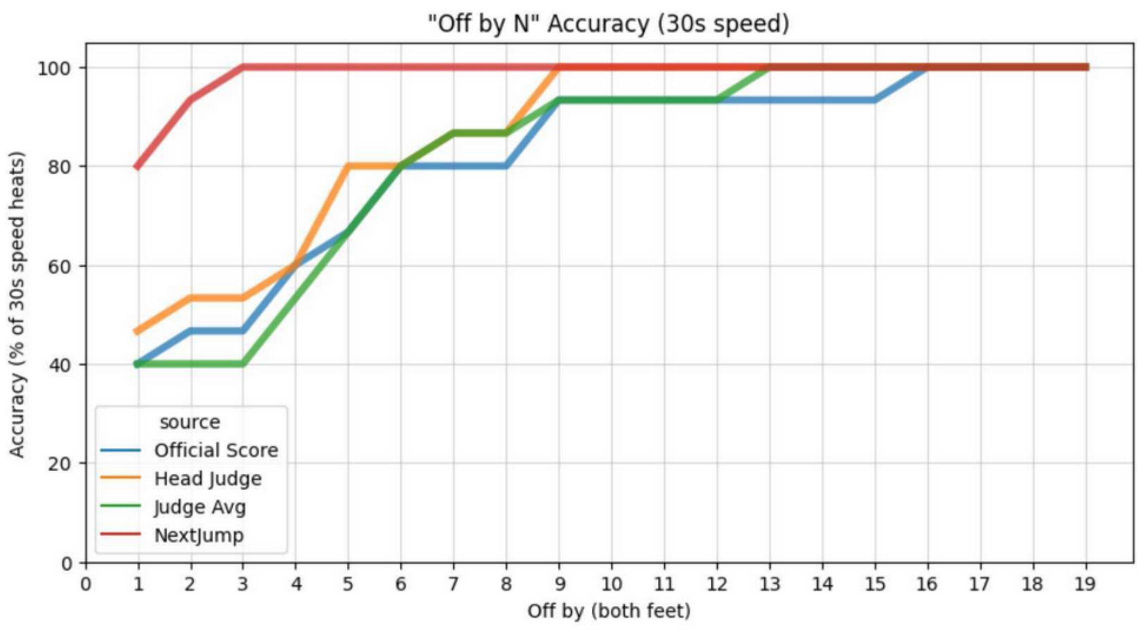


Figure A.2: Comparison of avg scores given to a jumper, compared to the effective score. Results are from 2024 AMJRF nationals.

without needing to cut out the different skills. Finally predicting the skill, which can be broken down into multiple, parallel runnable sections.

1. Jumper localization
2. Action segmentation, start/end of skill
3. Predict the effective skill
   (a) Predicting the level
   (b) Predict skill (power/gymnestic - pushup, split, cartwheel)
   (c) Predict turner involvement (cross, EB, TS)
   (d) Predict multiple (single, double, triple)

A.2.5. Jumper localization

Many research towards image recognition has been done (Zhengxia et al., 2019). The best models mostly utilize Convolutional Neural Networks pertaining spacial information in the image (Zaidi et al., 2021). In their paper, they compare some recent models for object detection such as YOLO(v4), CenterNet, SSD, EfficientDet-D2, each using some backbone architecture like VGG-16, AlexNet, GoogleNet or lightweight models such as ShuffleNet or MobileNet (all using CNN's). Some of them being real-time models (fps > 30). The goal of localizing the jumper is to center the athletes in the middle of the screen/video. Bharadiya (2023) elaborates that the position of objects in images doesn't really matter, but their are no clear statements about the size of objects. It could be that a jumper takes up 80 percent of

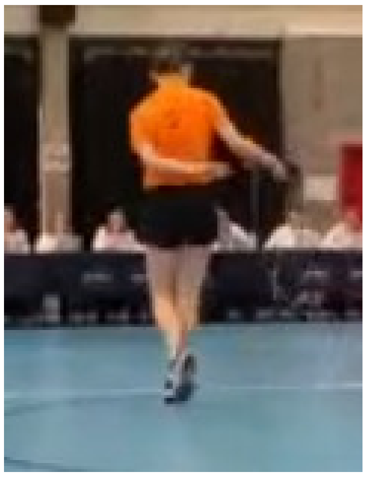


Figure A.3: Jumper wrapping the rope, SR

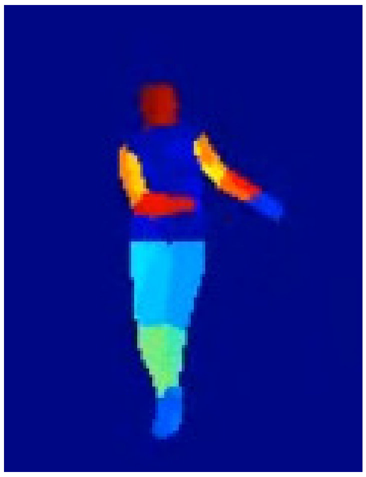


Figure A.4: Jumper wrapping the rope, denseposed

the screen, while moments later he moved backwards and only takes up 30 percent of the video. Instincts tell us that centered and scaled data will work better later.
One of these models can be taken as a base, using transfer learning[11] to fine-tune the results to localize the jumper.
To improve localization, video object segmentation or video instance segmentation can be used. Gao et al. (2022) lists some object segmentation models, like SwiftNet H. Wang et al. (2021) using ResNet18 as good and quick model. Other possibilities would be Cutie (Cheng et al., 2023), DensePose (see fig [A.3, A.4]) (Güler et al., 2018) Densepose seems to be able to give the bounding boxes of the main poses detected. Perhaps just a convex hull and some padding will be enough for smart cropping and training a network from scratch is not needed. However, a local try-out (without boxes) was rather slow, 1.2 fps on GPU within a Docker container, on a laptop. As jumpers don't move that much most of the time, skipping some frames

[11]Concept transfer learning explained in (Bharadiya, 2023)

and smoothing out the prediction over the rest of the sequence or decreasing the amount of skipped frames when movement is detected, can speed up the localization.
Even when Densepose isn't used, smoothing can still be applied to the guessed box as a video is basically a sequence of images.

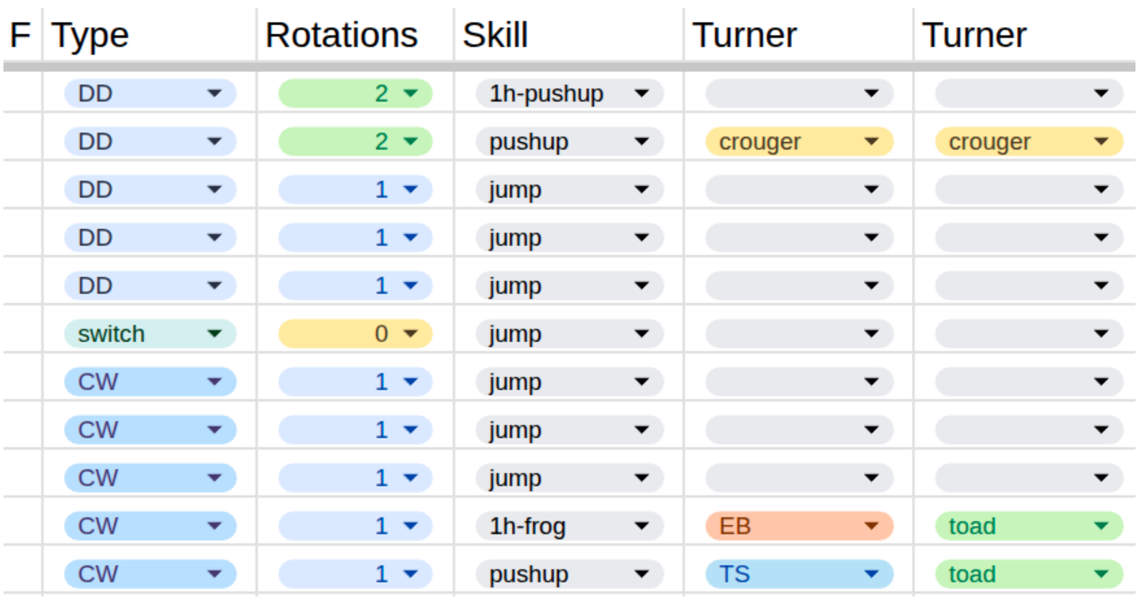
A.2.6. Video action segmentation

When the athletes are cropped, videos need to be split[12] in (sub)skills, because splitting new videos manually takes to much time and is impractical on competitions. A model like LTContext Zhou et al. (2023) or others[13] would be appropriate. Just like the localization, denseposing, extracting poses, foreground, background...would improve the segmentation.
The model for generating skill snapshots will be useful for judging and subsequently labeling data. Rather than replaying the video, judges can just sequentially go through each trick one at a time to assign, annotate or validate the predicted skill.

A.2.7. Skill recognition

The final step would be to recognize the total skill in a freestyle video. Yin et al. (2024) did a general HAR survey, also focused on team sports in which they describe the evolution from normal CNN architectures, to recurrent neural networks for time sequences, remembering context, like the Long Short Term Memory model (LSTM). Then combining CNN output into LSTM or even implementing a convolutional filter in the memory cell (Shi et al., 2015).
Y. Wang et al. (2019) investigated an improvement for current convolutional or recurrent models. They found that LSTM memory cells were too simple to contain higher-order complexities. As a result, they designed a memory in memory component, to replace the previous cell, which could predict actions on complexer data sets. This quickly formed the name Memory in Memory, MIM. Later, Z. Lin et al. (2020) used a self-attention memory cell inside the convLSTM that can memorize global aspects in time and space. With about 35% the number of parameters compared to Wang's MIM-model, SAM achieved a similar score as on the moving MNIST dataset, but faster. Although the focus of these papers was predicting future actions, the output can be transformed into a classification model, rather than a prediction model.
Another approach would be using transformers as a described possibility in Yin et al. (2024), namely ViT-TAD model by Yang et al. (2023), Swin transformer Liu, Lin et al. (2021) or VideoMAE v2 by L. Wang et al. (2023) seem good options, but further research/try-outs will be required to ensure the transfer models can predict actions.

[12]Not real/physical splits, but rather labels/annotations for where to split
[13]Using sources like paperswithcode/action-segmentation



F	Type	Rotations	Skill	Turner	Turner
	DD	2	1h-pushup		
	DD	2	pushup	crouger	crouger
	DD	1	jump		
	DD	1	jump		
	DD	1	jump		
	switch	0	jump		
	CW	1	jump		
	CW	1	jump		
	CW	1	jump		
	CW	1	1h-frog	EB	toad
	CW	1	pushup	TS	toad

Figure A.5: Representation of a skill-matrix used for Doube Dutch

For reference, NextJump uses a CNN - MobileNetv4, (Qin et al., 2024) and a transformer to analyze the full sequence and count. So using the convLSTM, SAM, MobileNet, or a transformer brings us to a better definition or example of what exactly we want to predict.

A.2.8. Skill-matrix - complexity & levels - towards model accuracy

A basic explanation of skills with some examples was given earlier. In order to recognize all skills, they should be worked out as good as possible. This section explains the composition of the skill-matrix, in order to give a better understanding on the total accuracy of the models later on. As described earlier, skills come with many transitions, take the first skill-matrix representation in figure A.5 to better understand skills and transitions.
Type: You have four styles of turning in double dutch; the normal way or double dutch (DD), reversing the rope rotation, sort of like backwards called irish dutch (irish), the chinese way of turning, chinese wheel (CW) or only using one rope or the two combined as a single rope, called single dutch or one rope.
Rotations: The amount of ropes passing underneath the athlete in one jump.
Turner: There are two turners, so two columns, where each turner can execute a turner involvement. Examples are EB, toad or crouger. The cross is in most cases performed by both turners, otherwise the ropes are tangled.
Skills: Mostly powers or gymnastics, but could also be footwork [14]. Further distinction or organizing can happen as variation and transitions of powers and gymnastics exists. A recap of different skills would be: pushups, splits, crabs, frogs, swift, cartwheel, salto, webster, suicide, handsprings, round offs ...Depending on the ex-

[14]Footwork will not be further specified in this paper

act power or gymnastic, certain characteristics or transitions could be applied:

- one handed
- one or two feet push-off (e.g. frog vs high frog, salto vs webster, suicide one vs two legged push-off)
- turntable [15]
- full body rotation [16]
- consecutive [17], e.g. frog after frog

Applying these transitions, characteristics allows for more variation, which gets you more points. Repeated skills doesn't get you points. Repetitions are only defined by the skills in the ropes, the speed of the rope (rotations) and the type of turning [18].

The skill-matrix is subject to change over time, initial skills not fitting the matrix will be left out for the MVP. Other than skills, jumpers and turners can switch with each other, most of these are also omitted in the MVP. Each column in the skill-matrix-example 5 can only contain one answer, for which softmax can be used, while between multiple columns, no relation is required and can be predicted separately. This can be solved using a multi-branch output as described in by (Coulibaly et al., 2022)). This can result in a guessed skill being partially correct, e.g. turner correct, but wrong rotational amount.
On another note, Guo et al. (2017) explain that softmax isn't always a good indicator of confidence. They mention that when a model has good calibration, the accuracy should align with the confidence of the prediction. E.g. when you predict a skill has 80% chance being a push-up, then this prediction or 80% of the predicted push-ups should be correct. Predictions using softmax tend to be overconfident.

A.2.9. Group activity

As DD3 is a group activity, problems could arise while trying to detect skills. However, the hypothesis is that a DD3 freestyle always acts as one unit, thus not really requiring much special attention. This could pose a problem when adapting SR to SR2 where two individuals are not exactly one unit. Some further research can be done in models like stagNet (Qi et al., 2020) to improve, incorporate this idea.

[15]On way of turning your bodyposition, can be done per quarter e.g. quarter turntable push-up
[16]Other way of turning your body, requires full turns
[17]Consecutive handstands are considered harder and thus get you extra points, not the case with push-up
[18]No difference will be made between irish turning and double dutch. Also, only the first skill in single dutch counts

A.2.10. Unknown/Unusual skills

Unknown skills or special cases pose a problem. That's why the skill-matrix needs to defined in such a way that new combinations can fit the matrix as much as possible and/or in combination with zero-shot learning. (Sort of marking unique skills as 'I don't know' so that others new/unique ones will also be marked as 'I don't know') Unusual skills on the other hand should be incorporated into the implementation. Earlier an example was given about knocking 3 times on the door. A more concrete example would be turntables, which are mainly performed using a crab or push-up, but also seen with a frog or a split. Turntables even have the potential to be combined with a swift. Omitting turntable frogs or splits in the train dataset can test this the ability to perform on unusual skills.

A.2.11. Summary literature

The PoC needs to localize the jumper [19], segment actions, and using labeled splits or guessed ones to predict (sub)skills. Predicted skills will then be mapped to their corresponding level.

A.3. Methodology

After some more research about transformers and how to apply them for skill recognition and another non-transformer models besides convLSTM & SAM to predict skills, the build process can start. Using Canva, a user story map is created to effectively see which tasks need to be done.

A.3.1. General & label location

- overview videos: navigate, filter, rename
- view video info, could have edit
- label inappropriate/blurry moments (idea: could be used, rather not)
- label passage/empty (livestream/wait) moments (idea: no skills)
- label jumper location
- Statistics of the general data distribution

A.3.2. Predict location

- visualize jumpoer location predictions
- edit borders from new predictions
- visualize biggest localization mistakes (within video)

[19]which can use fine-tuned pre-trained models

- visualize biggest localization mistakes (over all videos)
- localization model statistics
- data augmentation (if needed)

A.3.3. Label action segments + label skills

- Label video action segment
- Loop-replay action segment
- Label segmented skills names
- mark false skills
- skills to level mapper (should have)
- search skills by name (could have)
- mark execution scale (could have)
- skilldistribution statistics (+/- distributionmatrix)

A.3.4. Predict action segments

- visualize & compare AI segmented skills
- visualize AI segmented skills from new videos (with confidence levels?)

A.3.5. Predict skills

- model predicts levels (1-8 classes)
- model predicts variation element (4/6 classes)
- model predicts skill
- add judge scores to freestyles
- visualize & compare AI labeled skills & compare with total (judge)score
- validate AI recognized skills & save as new training input
- label AI recognized skills as new or ”i don’t know”
- action segmentation stats
- skillrecognition stats

A.3.6. Bis

- Language
- add competition jump order (easy recordings)
- stats competition
- insert multiple judge systems

A.3.7. Answering sub-questions

Most answers on the research (sub-)questions are woven into the literature, while others need to wait on the proof of concept to be fully answered. While labeling skills an answer can be given to question A.1.2 acceptable results. The effective usage of models and architectures A.1.2 and the educated guess about the data-amount needed to achieve better results A.1.2 will be answered soon

A.3.8. Training & Hardware

Working with videos alone requires a lot of resources, let alone training on the data with a normal laptop. It is best to work with one or more GPUs to aid the research process. In between, calculations can be made to estimate how long training sessions will take. This also gives a future reference for other computer vision concepts.

A.4. Expected results

Localization of the jumper shouldn't pose an issue. It's would be safe to say that a quick progression towards video action segmentation can be made. Even if segmented actions do not overlap with their actual skills, it's not that hard of a block for the skill recognition part. It would just mean, that skillrecognition can not be applied on new unsegmented videos. It's expected that SAM will prove to give better results than his base convLSTM model and a transformer probably even better if correctly integrated.
On the long run, it's expected to surpass judges in accuracy.
To answer the questions asked in the introduction a list of unknown or potential answers.

- How will the model be built? –> unknown
- What would be the main structure of the model? –> unknown
- Which human activity recognition examples can be used or altered as the base of the model? –> aligns with modelselection
- When are AI-recognitions acceptable to potentially use on competitions? –> when they score equally as good as a judge and can flag or anticipate unknowns.

- How much data is expected to increase the accuracy off the Judge. –> we'll know later
- How can we use the AI-Judge to improve judges? –> correcting/assisting them.
- What needs to changed to a working model, to apply it on other judge-sports such as gymnastics, synchronized swimming... –> define a skill-matrix & data

A.5. Expected conclusion

The AI-Judge can improve fairness of judging on competitions by annotating skills and, when expanded, giving confidence levels to the skills it predicted. Using AI-Judge, more transparency towards scores can increase competitiveness or enable the public to better understand the scores. Even using the model to display snap-shots of freestyles along the predicted skill and confidence score in total.
This AI-Judge can be extended towards other disciplines or different sports like acro, figure skating, gymnastics or dressage, to achieve similar results.
The next steps will be extending freestyle judging into DD4, SR2 and special skills.

B
Display of skills and turners

In this appendix, skills will be displayed to give an example what exactly needs to be recognized. The focus of the first section is showing some subdisciplines of jump rope. The second part involves displaying performed skills and turners in Double Dutch.

B.1. Jump Rope Disciplines

This section will focus on showing images of subdisciplines to make clear what entails each discipline.

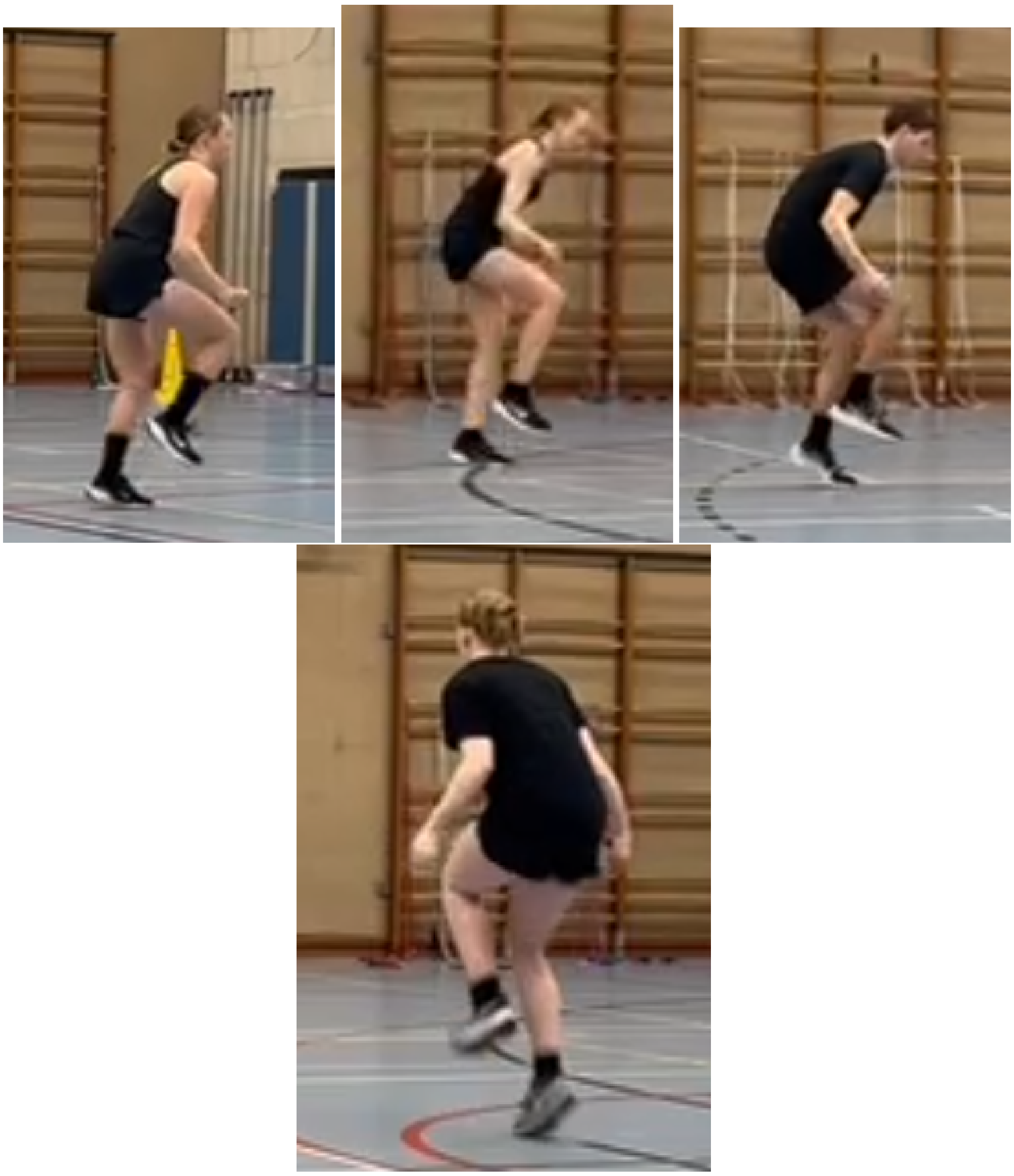


Figure B.1: Jumpers performing single rope speed.

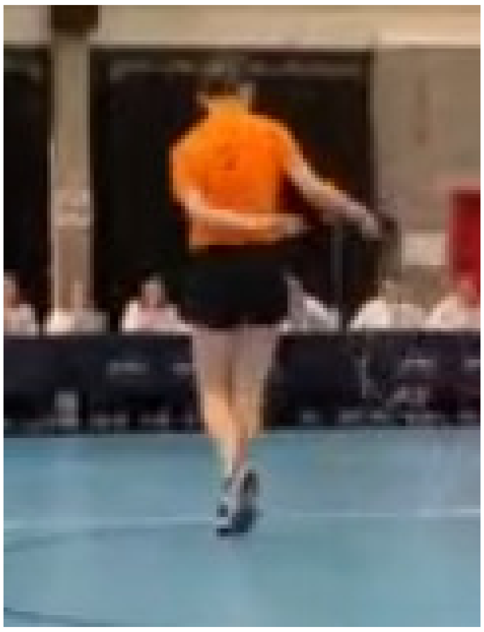


Figure B.2: Athlete in a Single Rope routine doing a wrap.

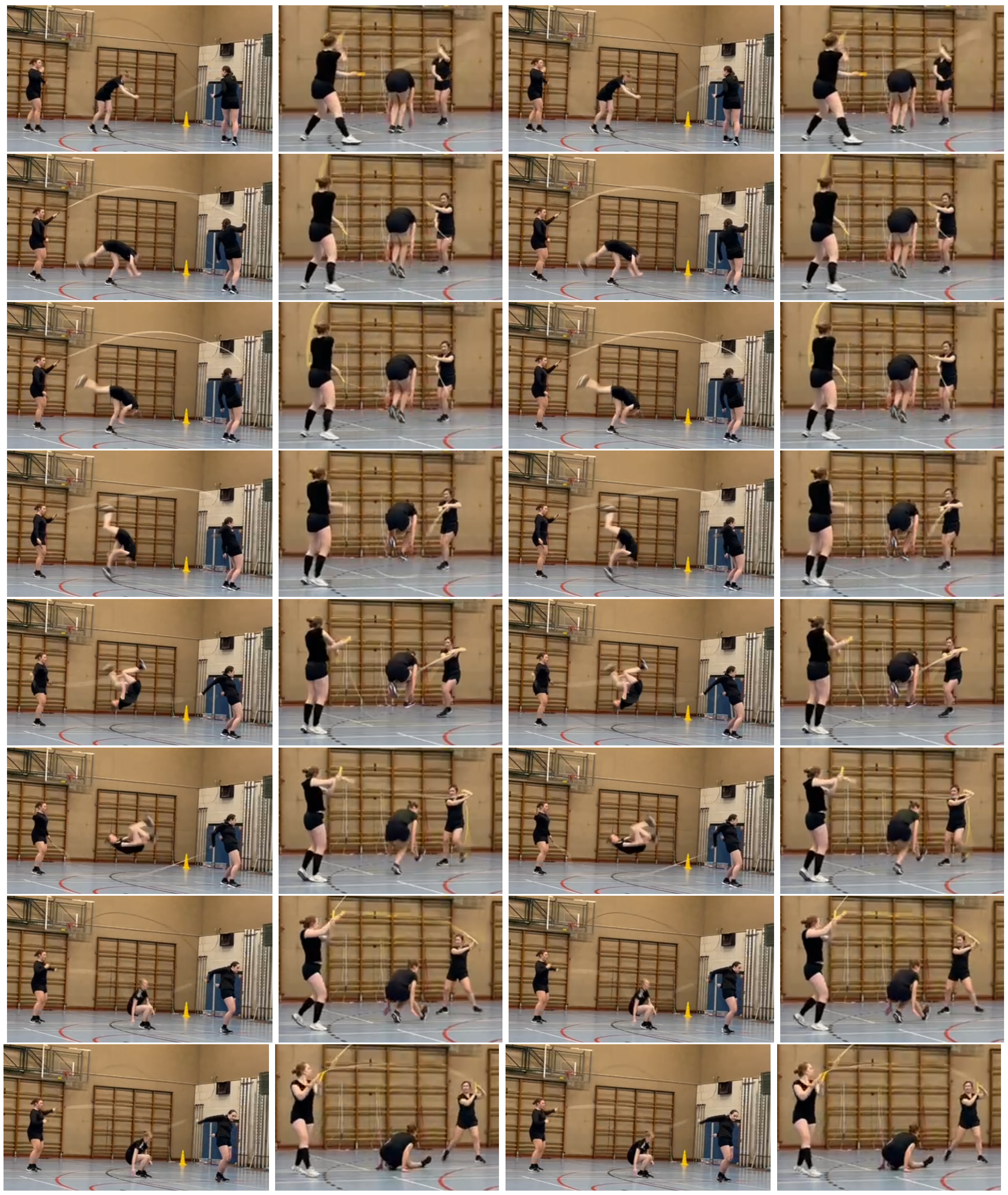


Figure B.3: Display of 4 skills (currently only 2, 2 others will be added later)
1: A flip while with double under and a TS restriction.
2: A double under with two crosses, while the jumper lands in a crab position.
3: Currently a duplicate of 1
4: Currently a duplicate of 2

C
Code snippets

In this appendix, code snippets to prevent code saturation in between text.
(Code example will be deleted in the final submit)

C.1. Localization

C.2. Segmentation

C.3. Recognition

```python
import pandas as pd
```

Listing C.1

```
DefaultMaxPool = functools.partial(
    keras.layers.MaxPool2D,
    pool_size=(3,3), strides=(2,2), padding="same")

def get_googlenetsmall_model(input_shape, num_classes,
    use_batch_norm=True, **kwargs):
    model = keras.Sequential(**kwargs)
    model.add(keras.layers.Input(shape=input_shape))
    model.add(DefaultConv(filters=24, kernel_size=(7,7),
        strides=(2,2),  padding='same'))
    if use_batch_norm:
        model.add(keras.layers.BatchNormalization())
    model.add(DefaultMaxPool())
    model.add(DefaultConv(filters=32))
    model.add(DefaultConv(filters=48, kernel_size=(3,3)))
    if use_batch_norm:
        model.add(keras.layers.BatchNormalization())
    model.add(DefaultMaxPool())

    model.add(InceptionModule(filters11=32, filters33_reduce=48,
        filters33=64,
        filters55_reduce=8, filters55=16, filters_pool_proj=16,
        use_batch_norm=use_batch_norm))
    model.add(InceptionModule(filters11=64, filters33_reduce=64,
        filters33=96,
        filters55_reduce=16, filters55=48, filters_pool_proj=32,
        use_batch_norm=use_batch_norm))
    model.add(DefaultMaxPool())

    # ... see part 2
```

Listing C.2: GoogleNet implementation part 1

```
def get_googlenetsmall_model(input_shape, num_classes,
    use_batch_norm=True, **kwargs):
    # ... see part 1

    model.add(InceptionModule(filters11=96, filters33_reduce=48,
        filters33=104,
        filters55_reduce=8, filters55=24, filters_pool_proj=32,
        use_batch_norm=use_batch_norm))
    model.add(InceptionModule(filters11=80, filters33_reduce=56,
        filters33=224,
        filters55_reduce=12, filters55=32, filters_pool_proj=32,
        use_batch_norm=use_batch_norm))
    model.add(InceptionModule(filters11=64, filters33_reduce=64,
        filters33=256,
        filters55_reduce=12, filters55=32, filters_pool_proj=32,
        use_batch_norm=use_batch_norm))
    model.add(InceptionModule(filters11=56, filters33_reduce=64,
        filters33=144,
        filters55_reduce=16, filters55=32, filters_pool_proj=32,
        use_batch_norm=use_batch_norm))
    model.add(InceptionModule(filters11=128, filters33_reduce=80,
        filters33=160,
        filters55_reduce=16, filters55=64, filters_pool_proj=64,
        use_batch_norm=use_batch_norm))

    model.add(DefaultMaxPool())
    model.add(InceptionModule(filters11=128, filters33_reduce=80,
        filters33=160,
        filters55_reduce=16, filters55=64, filters_pool_proj=64,
        use_batch_norm=use_batch_norm))
    model.add(InceptionModule(filters11=192, filters33_reduce=92,
        filters33=192,
        filters55_reduce=24, filters55=64, filters_pool_proj=64,
        use_batch_norm=use_batch_norm))
    model.add(keras.layers.GlobalAveragePooling2D())
    model.add(keras.layers.Dropout(0.4))
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(units=256, activation="relu"))
    model.add(keras.layers.Dense(units=num_classes,
        activation='sigmoid'))

    return model
```

Listing C.3: GoogleNet implementation part 2

```python
import functools
import keras

DefaultConv = functools.partial(
    keras.layers.Conv2D, kernel_size=(3, 3), strides=(2, 2),
    padding="same", activation="relu",
        kernel_initializer="he_normal")

DefaultMaxPool = functools.partial(
    keras.layers.MaxPool2D,
    pool_size=(3,3), strides=(2,2), padding="same")

def get_model(modelinfo: dict, **kwargs):
    dim = modelinfo['dim']
    model = keras.Sequential(**kwargs)
    mobilenetv3small = keras.applications.MobileNetV3Small(
        input_shape=(dim,dim,3),
        include_top=False,
        weights="imagenet",
        dropout_rate=0.2,
        pooling='avg',
        name="MobileNetV3Small",
    )
    mobilenetv3small.trainable = False
    model.add(mobilenetv3small)
    model.add(keras.layers.Dense(units=512, activation="relu"))
    model.add(keras.layers.Dense(units=128, activation="relu"))
    model.add(keras.layers.Dense(units=4, activation='sigmoid'))

    return model
```

Listing C.4: Usage of mobilenet for full box predictions

```python
# To long to properly include in the paper
#
    https://github.com/mikeddecker/judge/blob/main/computervision/localizor_with_s
```

Listing C.5: Localizor with strats

```python
def calculate_splitpoint_values(videoId: int, frameLength:int,
    df_Skills:pd.DataFrame, fps:float, Nsec_frames_around=1/6):
"""Creates a dataframe: 'videoId', 'frameNr', 'splitpoint'
Where splitpoint is the value 0 -> 1 whether the video needs to be
    split at that point or not"""
splitpoint_values = {
    'videoId' : [videoId for _ in range(frameLength)],
    'frameNr' : range(frameLength),
    'splitpoint' : [0 for _ in range(frameLength)],
}

frames_around_splitpoint = round(Nsec_frames_around * fps)
for _, skillrow in df_Skills.iterrows():
    frameStart = skillrow["frameStart"]
    frameEnd = skillrow["frameEnd"]

    currentFrameStart = frameStart - frames_around_splitpoint
    currentFrameEnd = frameEnd - frames_around_splitpoint
    while currentFrameStart < frameStart + frames_around_splitpoint:
        framesApart = abs(currentFrameStart - frameStart)
        splitvalue = 1 - (framesApart/frames_around_splitpoint) ** 2
        splitvalue *= splitvalue

        currentFrameStart += 1
        currentFrameEnd += 1

        splitpoint_values['splitpoint'][currentFrameStart] =
            splitvalue
        if currentFrameEnd < frameLength:
            splitpoint_values['splitpoint'][currentFrameEnd] =
                splitvalue

return pd.DataFrame(splitpoint_values)
```

Listing C.6: call-splitpoint-calculation

```
df = calculate_splitpoint_values(
    videoId=videoId,
    frameLength=frameLength,
    df_Skills=self.Skills[self.Skills['videoId'] == videoId],
    fps = row["fps"]
)
```

Listing C.7: call-splitpoint-calculation

```
# ConfigHelper.py
def get_discipline_DoubleDutch_config(include_tablename=True):
    config = {
        "Type" : ("Categorical", "Type"), # Will be textual
            representions
        "Rotations" : ("Numerical", 0, 8, 1), # min, max, step
        "Turner1": ("Categorical", "Turner"),
        "Turner2": ("Categorical", "Turner"),
        "Skill" : ("Categorical", "Skill"),
        "Hands" : ("Numerical", 0, 2, 1), # 0 for al salto types
        "Feet" : ("Numerical", 0, 2, 1),
        "Turntable" : ("Numerical", 0, 4, 0.25), # Per quarter (but
            still integers)
        "BodyRotations" : ("Numerical", 0, 2, 0.5),
        "Backwards" : ("Boolean"),
        "Sloppy" : ("Boolean"),
        "Hard2see" : ("Boolean"),
        "Fault" : ("Boolean"),
    }
    if include_tablename:
        config["Tablename"] = "DoubleDutch"
    return config
```

Listing C.8: ConfigHelper skill configuration for labeling aspects of a skill, devided in Categorical, Numerical and Boolean output values of skills. All values of the numerical output are integers, which is why you need to multiply by the step in order to get the actual numerical representation.

```python
import keras
import pandas as pd
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import sys
sys.path.append('..')
from api.helpers import ConfigHelper
```

Listing C.9: imports-ViViT

```python
def mlp(x, hidden_units, dropout_rate):
    for units in hidden_units:
        x = keras.layers.Dense(units,
            activation=keras.activations.gelu)(x)
        x = keras.layers.Dropout(dropout_rate)(x)
    return x
```

Listing C.10: ViViT: multi-layer-perceptron

```python
def __init__(self, patch_size):
    super().__init__()
    self.patch_size = patch_size

def call(self, images):
    input_shape = keras.ops.shape(images)
    batch_size = input_shape[0]
    timestep = input_shape[1]
    height = input_shape[2]
    width = input_shape[3]
    channels = input_shape[4]
    num_patches_h = height // self.patch_size
    num_patches_w = width // self.patch_size

    def create_single_timepatch(video_input):
        patches = keras.ops.image.extract_patches(video_input,
            size=self.patch_size)
        patches = keras.ops.reshape(
            patches,
            (
                num_patches_h * num_patches_w * timestep,
                self.patch_size * self.patch_size * channels,
            ),
        )

        return patches

    patches = tf.map_fn(create_single_timepatch, images)

    return patches

```

Listing C.11: ViViT: Time Patches

```
def get_model(modelinfo):
    inputs = keras.Input(shape = (modelinfo['timesteps'],
        modelinfo['dim'], modelinfo['dim'], 3))
    patches = TimePatches(modelinfo['patch_size'])(inputs)
    num_patches = (modelinfo['dim'] // modelinfo['patch_size']) ** 2
    encoded_patches = TimePatchEncoder(num_patches,
        modelinfo['timesteps'], modelinfo['dim_embedding'])(patches)
    print(”shape of encoded_patches”, encoded_patches.shape)

    # Create multiple layers of the Transformer block.
    for _ in range(modelinfo['encoder_blocks']):
        # Layer normalization 1.
        x1 =
            keras.layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
        attention_output = keras.layers.MultiHeadAttention(
        num_heads=modelinfo['num_heads'],
            key_dim=modelinfo['dim_embedding'], dropout=0.1
        )(x1, x1)
        # Skip connection 1.
        x2 = keras.layers.Add()([attention_output, encoded_patches])
        # Layer normalization 2.
        x3 = keras.layers.LayerNormalization(epsilon=1e-6)(x2)
        x3 = mlp(x3, hidden_units=[modelinfo['dim_embedding'] ** 2,
            modelinfo['dim_embedding']], dropout_rate=0.1)
        # Skip connection 2.
        encoded_patches = keras.layers.Add()([x3, x2])

    # Create a [batch_size, projection_dim] tensor.
    representation =
        keras.layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
    representation = keras.layers.Flatten()(representation)
    representation = keras.layers.Dropout(0.3)(representation)

    features = mlp(representation,
        hidden_units=modelinfo['mlp_head_units'], dropout_rate=0.3)

    ...

```

Listing C.12: ViViT: get model ViViT (keras), without output layer (part 1) (Possible bugfixes done using OpenAI (2025))

```python
# ... (this is the output layer of segmentation)
classes = modelinfo['timesteps']
outputs = keras.layers.Dense(classes,
    activation='softmax')(features)

return keras.Model(inputs=inputs, outputs=outputs)
```

Listing C.13: ViViT output segmentation using features of code fragment C.12

```python
predictions = np.array(predictions)
predictions_bigger_than_split_threshold = np.where(predictions >
    split_threshold, predictions, 0)
p_split = predictions_bigger_than_split_threshold
window_size = int(fps // 3)
predictions_argMax_in_window = [s - window_size +
    np.argmax(p_split[max(0, s-window_size):min(frameLength,
    s+window_size)]) for s in range(frameLength)]
predictions_splitmoments = np.where(predictions > split_threshold,
    predictions_argMax_in_window, 0)
predictions_splitmoments = np.unique(predictions_splitmoments)

distances = predictions_splitmoments[1:] -
    predictions_splitmoments[:-1]
predictions_splitmoments = predictions_splitmoments[1:]
predictions_splitmoments =
    predictions_splitmoments[np.where(distances < window_size // 3,
    False, True)]
predictions_splitmoments = [int(g) for g in
    predictions_splitmoments]
```

Listing C.14: Code which filters splitpoints from the raw predicted splitpoint values to frame numbers.

```python
dd_config = ConfigHelper.get_discipline_DoubleDutch_config()
outputs = {}
for key, value in dd_config.items():
    if key == "Tablename":
        continue
    if value[0] == "Categorical":
        tablename = "skill"
        match (key):
            case 'Skill':
                tablename = 'skills'
            case 'Turner1' | 'Turner2':
                tablename = "turners"
            case 'Type':
                tablename = 'types'
        classes = int(df_table_counts.iloc[0][tablename] + 1) #
            Because of indexing
        outputs[key] = keras.layers.Dense(classes,
            activation='softmax', name=key)(features)
    else:
        outputs[key] = keras.layers.Dense(1, activation='sigmoid',
            name=key)(features)

return keras.Model(inputs=inputs, outputs=outputs)
```

Listing C.15: Keras output layers for skills

```python
def create_pytorch_skill_output_layers(lastNNeurons, balancedType, df_table_counts):
    dd_config = get_discipline_DoubleDutch_config()
    output_layers = torch.nn.ModuleDict()

    for key, value in dd_config.items():
        if key == "Tablename":
            continue
        if value[0] == "Categorical":
            columnname = "skill"
            if key == 'Skill':
                columnname = 'skills'
            elif key in ['Turner1', 'Turner2']:
                columnname = "turners"
            elif key == 'Type':
                columnname = 'types'

            classes = int(df_table_counts.iloc[0][columnname] + 1) # Because of MysqlIndex starts from 1
            output_layers[key] = torch.nn.Linear(lastNNeurons, classes)
        else:
            output_layers[key] = torch.nn.Linear(lastNNeurons, 1)

    if balancedType == 'jump_return_push_frog_other':
        output_layers['Skill'] = torch.nn.Linear(lastNNeurons, 5)

    return output_layers
```

Listing C.16: PyTorch skill output layers, uses C.8

```
class MViT(nn.Module):
    def __init__(self, skill_or_segment:str, modelinfo:dict,
        df_table_counts:pd.DataFrame):
        super(MViT, self).__init__()
        self.modelinfo = modelinfo
        self.df_table_counts = df_table_counts
        self.isSkillModel = skill_or_segment == ”skills”

        input_shape = (3, modelinfo['timesteps'], modelinfo['dim'],
            modelinfo['dim'])
        self.mvit = models.video.mvit_v1_b(weights='DEFAULT')
        self.mvit = self.mvit.to('cuda').eval()

        for param in self.mvit.parameters():
            param.requires_grad = False

        self.mvit.head = torch.nn.Identity()  # This removes the top
            layer
        self.flatten = nn.Flatten()
        self.LastNNeurons = self._get_mvit_output(input_shape)

        if self.isSkillModel:
            self.output_layers = cre-
                ate_pytorch_skill_output_layers(lastNNeurons=self.LastNNeurons,
                balancedType=modelinfo['balancedType'],
                df_table_counts = self.df_table_counts)
        else:
            self.output_layer = cre-
                ate_pytorch_segmentation_output_layers(lastNNeurons=self.LastNNeur
                timesteps=modelinfo['timesteps'])


    def _get_mvit_output(self, shape):
        with torch.no_grad():
            input = torch.rand(1, *shape).to('cuda')
            output = self.mvit(input)
            output = self.flatten(output)
            return output.shape[1]
```

Listing C.17: Pytorch MViT implementation init, uses C.16 and C.18

```python
def forward(self, x):
    # Input shape: (batch_size, channels, timesteps, height, width)
    x = self.mvit(x)
    x = self.flatten(x)

    if self.isSkillModel:
        return forward_skill_output_layers(features=x,
            output_layers=self.output_layers)
    else:
        return forward_segmentation_output_layers(features=x,
            output_layer=self.output_layer)
```

Listing C.18: Pytorch MViT implementation of forward method, used by C.17

```python
def forward_skill_output_layers(features: torch.tensor,
    output_layers: dict[str, torch.nn.Module]):
    outputs = {}
    for key, layer in output_layers.items():
        if key in ['Skill', 'Turner1', 'Turner2', 'Type']:
            outputs[key] = layer(features)
        else:  # Regression outputs
            outputs[key] = torch.sigmoid(layer(features))

    return outputs
```

Listing C.19: PyTorch skill forward feeding

```
# Adapting the losses, as limiting to 10% can change occurences of
    faults, bodyrotations... a little
for key, value in config.items():
    value_counts_train =
        train_generator.BalancedSet[ConfigHelper.lowerProperty(key)].value_counts(
    value_counts_val =
        val_generator.Skills[ConfigHelper.lowerProperty(key)].value_counts(dropna=
    value_counts_combined = value_counts_train.add(value_counts_val,
        fill_value=0)

    maximum = value_counts_combined.max()

    weights = (maximum + maximum // 8 -
        value_counts_combined).pow(0.75)
    weights = weights / weights.mean()
    if value[0] == 'Categorical':
        weights.loc[0] = 0
    weights = weights.sort_index()

    w_all = torch.ones(value_counts_combined.index.max() + 1,
        dtype=torch.float32).to(device=device)
    for idx, w in weights.items():
        w_all[idx] = w
    w_all = (w_all + 1) ** 2

    print(”loss weights for”, key, w_all)
    if value[0] == 'Categorical':
        loss_fns[key] =
            torch.nn.CrossEntropyLoss(w_all).to(device=device)
    else:
        loss_fns[key] = lambda input, target:
            weighted_mse_loss(input=input, target=target,
            weight=w_all)
```

Listing C.20: Code calculating weights for the loss functions

```python
def weighted_mse_loss(input, target, weight):
        "https://discuss.pytorch.org/t/how-to-implement-weighted-mean-square-error
    return torch.sum(weight * (input - target) ** 2)
```

Listing C.21: Weighted MSE

D

Tables

videoId	**seconds**	**reason**
...	...	(other videos)
1435	0	
1445	0	
2285	0	
2295	0	
2305	0	
2315	0	
995	5	Schocks & turner out-of-view
1275	1	aerial half
1405	30	spectator
651	1	turner out of view
664	1.5	turner out of view
1202	5	turner out of view, recording itself
1178	2	turner out of view
1238	2	spectator before start routine
1244	0	stopped following, remained in view
1257	0	stopped following, remained in view
1275	1.5	half out of view
1339	8	spectator
1444	1.5	half out of view
2282	1	1 turner out of view, actual 3 sec no follow
2317	2	minor shocks
663	25	Out-of-view & shocks
1241	8	Out-of-view
1267	46	spectator
1271	24	Out-of-view
1273	8	Out-of-view
1326	6.5	spectator
avg	1.33	(seconds / 1m15)
avg_{if}	8.55	(seconds / 1m15)
Nr of videos not ok	21	15.6%
Total checked videos	135	(includes train videos)

Table D.1: Crop results manually checked on the amount of seconds where a crop is disturbing. The reason why it is disturbing is added in the reason column, this could be athletes out of view, unstable/earthquake like predictions...
Yellow indicated videos are not disturbing for recognition, even though the crops aren't perfect. Orange videos require more attention and shouldn't be used for recognition.
The avg_{if} indicates the average time in case a video has crops which aren't good.

	precision	recall	f1-score	support
Double Dutch	0.97	0.98	0.97	525
Single Dutch	1.00	0.25	0.40	4
Irish Dutch	0.00	0.00	0.00	0
Chinese Wheel	0.92	0.97	0.94	86
Transition	0.72	0.88	0.79	24
Snapperlike	0.90	0.63	0.75	30
accuracy			0.95	669
macro avg	0.75	0.62	0.64	669
weighted avg	0.95	0.95	0.95	669

Table D.2: First MViT class report for type

	precision	recall	f1-score	support
0	0.00	0.00	0.00	26
1	0.91	0.95	0.93	544
2	0.59	0.67	0.63	70
3	0.39	0.44	0.41	16
4	0.50	0.10	0.17	10
5	0.00	0.00	0.00	2
8	0.00	0.00	0.00	1
accuracy			0.86	669
macro avg	0.34	0.31	0.31	669
weighted avg	0.82	0.86	0.83	669

Table D.3: First MViT class report for rotations

	precision	recall	f1-score	support
normal	0.93	0.98	0.96	542
crouger	0.90	0.88	0.89	40
cross	0.80	0.58	0.67	57
cross BW	0.00	0.00	0.00	2
jump over cross BW	0.00	0.00	0.00	3
EB	0.57	0.36	0.44	11
toad	0.54	0.88	0.67	8
toad BW	0.00	0.00	0.00	1
EB toad	0.00	0.00	0.00	3
TS	0.00	0.00	0.00	0
inverse toad	0.00	0.00	0.00	0
elephant	0.00	0.00	0.00	0
crougercross	0.00	0.00	0.00	0
pinwheel	0.00	0.00	0.00	1
suicide	0.00	0.00	0.00	0
inverse crouger	0.00	0.00	0.00	0
flip	0.00	0.00	0.00	0
T-toad	0.00	0.00	0.00	0
MULTIPLE TURNERSKILLS	0.00	0.00	0.00	0
EB toad BW	0.00	0.00	0.00	0
L2-power-gym	0.00	0.00	0.00	0
L3-power-gym	0.00	0.00	0.00	0
L4-power-gym	0.00	0.00	0.00	0
UNKNOWN	0.00	0.00	0.00	0
jump-through	0.00	0.00	0.00	0
EB inverse toad	0.00	0.00	0.00	1
accuracy			0.91	669
macro avg	0.14	0.14	0.14	669
weighted avg	0.89	0.91	0.90	669

Table D.4: First MViT class report for turner1

	precision	recall	f1-score	support
normal	0.92	0.97	0.94	540
crouger	0.80	0.84	0.82	38
cross	0.79	0.60	0.68	55
cross BW	0.00	0.00	0.00	4
jump over cross BW	0.00	0.00	0.00	2
EB	0.67	0.31	0.42	13
toad	0.50	0.86	0.63	7
toad BW	0.00	0.00	0.00	2
EB toad	0.00	0.00	0.00	4
TS	0.00	0.00	0.00	0
inverse toad	0.00	0.00	0.00	0
elephant	0.00	0.00	0.00	0
crougercross	0.00	0.00	0.00	0
pinwheel	0.00	0.00	0.00	3
suicide	0.00	0.00	0.00	0
inverse crouger	0.00	0.00	0.00	0
flip	0.00	0.00	0.00	0
T-toad	0.00	0.00	0.00	0
MULTIPLE TURNERSKILLS	0.00	0.00	0.00	0
EB toad BW	0.00	0.00	0.00	0
L2-power-gym	0.00	0.00	0.00	0
L3-power-gym	0.00	0.00	0.00	0
L4-power-gym	0.00	0.00	0.00	0
UNKNOWN	0.00	0.00	0.00	0
jump-through	0.00	0.00	0.00	0
EB inverse toad	0.00	0.00	0.00	1
accuracy			0.90	669
macro avg	0.14	0.14	0.13	669
weighted avg	0.87	0.90	0.88	669

Table D.5: First MViT class report for turner2

	precision	recall	f1-score	support
jump	1.00	0.98	0.99	380
return from power	0.88	0.87	0.88	85
pushup	0.84	0.96	0.90	72
frog	0.96	0.90	0.93	61
split	0.94	1.00	0.97	15
SPAGAAT	0.00	0.00	0.00	0
crab	0.64	0.88	0.74	8
swift	0.00	0.00	0.00	0
mountainclimber	0.00	0.00	0.00	0
rad	0.00	0.00	0.00	2
rondat	0.40	0.40	0.40	5
handspring	0.43	0.60	0.50	5
kip	1.00	0.83	0.91	6
kopkip	0.50	1.00	0.67	1
flip	0.50	0.60	0.55	5
arabian	0.00	0.00	0.00	0
rol2kip	0.83	0.83	0.83	6
roll	1.00	0.67	0.80	3
leapfrog	0.00	0.00	0.00	0
suicide	0.89	0.89	0.89	9
flip-to-pushup	0.00	0.00	0.00	1
buddy-bounce	0.00	0.00	0.00	1
stut	1.00	1.00	1.00	2
footwork-kick	0.00	0.00	0.00	0
speed	0.00	0.00	0.00	0
footwork-open	0.00	0.00	0.00	1
footwork-knee	0.00	0.00	0.00	0
footwork-cancan	0.00	0.00	0.00	0
footwork-cross	0.00	0.00	0.00	0
UNKOWN	0.00	0.00	0.00	1
accuracy			0.93	669
macro avg	0.39	0.41	0.40	669
weighted avg	0.93	0.93	0.93	669

Table D.6: First MViT class report for skill

	precision	recall	f1-score	support
0	0.99	0.94	0.96	481
1	0.27	0.27	0.27	45
2	0.75	0.88	0.81	143
accuracy			0.88	669
macro avg	0.67	0.70	0.68	669
weighted avg	0.89	0.88	0.88	669

Table D.7: First MViT class report for hands

	precision	recall	f1-score	support
0	0.94	0.70	0.80	67
1	0.52	0.52	0.52	83
2	0.92	0.95	0.94	519
accuracy			0.87	669
macro avg	0.80	0.72	0.75	669
weighted avg	0.87	0.87	0.87	669

Table D.8: First MViT class report for feet

	precision	recall	f1-score	support
0	0.98	0.99	0.99	656
1	0.17	0.09	0.12	11
2	0.00	0.00	0.00	2
accuracy			0.97	669
macro avg	0.38	0.36	0.37	669
weighted avg	0.97	0.97	0.97	669

Table D.9: First MViT class report for turntable

	precision	recall	f1-score	support
0	1.00	1.00	1.00	666
1	0.00	0.00	0.00	2
2	0.00	0.00	0.00	1
accuracy			1.00	669
macro avg	0.33	0.33	0.33	669
weighted avg	0.99	1.00	0.99	669

Table D.10: First MViT class report for bodyRotations

	precision	recall	f1-score	support
0	0.99	1.00	1.00	660
1	0.80	0.44	0.57	9
accuracy			0.99	669
macro avg	0.90	0.72	0.78	669
weighted avg	0.99	0.99	0.99	669

Table D.11: First MViT class report for backwards

	precision	recall	f1-score	support
0	0.97	1.00	0.99	651
1	0.00	0.00	0.00	18
accuracy			0.97	669
macro avg	0.49	0.50	0.49	669
weighted avg	0.95	0.97	0.96	669

Table D.12: First MViT class report for sloppy

	precision	recall	f1-score	support
0	1.00	1.00	1.00	666
1	0.00	0.00	0.00	3
accuracy			1.00	669
macro avg	0.50	0.50	0.50	669
weighted avg	0.99	1.00	0.99	669

Table D.13: First MViT class report for hard2see

	precision	recall	f1-score	support
0	0.96	1.00	0.98	639
1	0.00	0.00	0.00	30
accuracy			0.96	669
macro avg	0.48	0.50	0.49	669
weighted avg	0.91	0.96	0.93	669

Table D.14: First MViT class report for fault

	precision	recall	f1-score	support
0	0.96	1.00	0.98	801.0
1	0.71	0.26	0.38	39.0
accuracy			0.96	840.0
macro avg	0.84	0.63	0.68	840.0
weighted avg	0.95	0.96	0.95	840.0

Table D.15: Fault class report after weighted losses

Bibliography

Abdullah, A. S., & Alsaif, K. I. (2023). Still rings movements recognition in gymnastics sport based on deep learning. *Wasit Journal for Pure sciences*, *2*(1), 207–216.

Alharthi, A. S., Yunas, S. U., & Ozanyan, K. B. (2019). Deep learning for monitoring of human gait: A review. *IEEE Sensors Journal*, *19*(21), 9575–9591. https://doi.org/10.1109/JSEN.2019.2928777

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., & Schmid, C. (2021). Vivit: A video vision transformer. https://doi.org/https://doi.org/10.48550/arXiv.2103.15691

Bharadiya, J. P. (2023). Convolutional neural networks for image classification. *International Journal of Innovative Science and Research Technology*. https://doi.org/10.5281/zenodo.7952031

Bora, J., Dehingia, S., Boruah, A., Chetia, A. A., & Gogoi, D. (2023). Real-time assamese sign language recognition using mediapipe and deep learning. *Procedia Computer Science*, *218*, 1384–1393. https://doi.org/https://doi.org/10.1016/j.procs.2023.01.117

Cheng, H. K., Oh, S. W., Price, B., Lee, J.-Y., & Schwing, A. (2023). Putting the object back into video object segmentation. https://doi.org/https://doi.org/10.48550/arXiv.2310.12982

Coulibaly, S., Kamsu-Foguem, B., Kamissoko, D., & Traore, D. (2022). Deep convolution neural network sharing for the multi-label images classification. *Machine Learning with Applications*, *10*, 100422. https://doi.org/10.1016/j.mlwa.2022.100422

Deepseek. (2025). *Deepseek (april 2025 free version) [large language model]*. Retrieved April 1, 2025, from https://deepseek.com

Diwakar & Raj, D. (2022). Recent object detection techniques: A survey. *International Journal of Image, Graphics and Signal Processing*, *14*(2), 47–60. https://doi.org/10.5815/ijigsp.2022.02.05

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., & Feichtenhofer, C. (2021). Multiscale vision transformers. https://doi.org/10.48550/ARXIV.2104.11227

Fujitsu. (2023). *Fujitsu and the international gymnastics federation launch ai-powered fujitsu judging support system for use in competition for all 10 apparatuses*. Retrieved May 15, 2025, from https://www.fujitsu.com/global/about/resources/news/press-releases/2023/1005-02.html

Gao, M., Zheng, F., Yu, J. J. Q., Shan, C., Ding, G., & Han, J. (2022). Deep learning for video object segmentation: A review. *Artificial Intelligence Review*, *56*(1), 457–531. https://doi.org/10.1007/s10462-022-10176-7

Güler, R. A., Neverova, N., & Kokkinos, I. (2018). Densepose: Dense human pose estimation in the wild. https://doi.org/https://doi.org/10.48550/arXiv.1802.00434

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. https://doi.org/10.48550/ARXIV.1706.04599

Gymfed. (2017). *Offiële records rope skipping* [File updated in 2024]. Retrieved December 9, 2024, from https://www.gymfed.be/nieuws/officiele-records-rope-skipping

Heiniger, S., & Mercier, H. (2018). Judging the judges: A general framework for evaluating the performance of international sports judges. https://doi.org/10.48550/ARXIV.1807.10055

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for mobilenetv3. https://doi.org/https://doi.org/10.48550/arXiv.1905.02244

IJRU. (2023a). *Live | world jump rope championships 2023 | day 1*. International Jump Rope Union. Retrieved December 18, 2024, from https://www.youtube.com/watch?v=EqczXTJliyc

IJRU. (2023b). *Live | world jump rope championships 2023 | day 8*. International Jump Rope Union. Retrieved December 18, 2024, from https://www.youtube.com/watch?v=PnHdv-a4yWA

IJRU. (2024a). *American records*. Retrieved December 20, 2024, from https://www.amjrf.com/page/show/8542453-us-national-records

IJRU. (2024b). *World records*. Retrieved December 9, 2024, from https://ijru.sport/world-records

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., & Zisserman, A. (2017). The kinetics human action video dataset. https://doi.org/https://doi.org/10.48550/arXiv.1705.06950

Khanam, R., & Hussain, M. (2024). Yolov11: An overview of the key architectural enhancements. https://doi.org/https://doi.org/10.48550/arXiv.2410.17725

Laue, R. (1999). *Rope skipping records from all over the world*. Retrieved December 9, 2024, from https://eherber.home.xs4all.nl/ropeskip/recordsm.htm

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2014). Microsoft coco: Common objects in context. https://doi.org/https://doi.org/10.48550/arXiv.1405.0312

Lin, Z., Li, M., Zheng, Z., Cheng, Y., & Yuan, C. (2020). Self-attention convlstm for spatiotemporal prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(07), 11531–11538. https://doi.org/10.1609/aaai.v34i07.6819

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. https://doi.org/https://doi.org/10.48550/arXiv.2103.14030

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., & Hu, H. (2021). Video swin transformer. https://doi.org/https://doi.org/10.48550/arXiv.2106.13230

Luqman, H., & Elalfy, E. (2022). Utilizing motion and spatial features for sign language gesture recognition using cascaded cnn and lstm models. *Turkish Journal of Electrical Engineering and Computer Sciences*, *30*(7), 2508–2525. https://doi.org/10.55730/1300-0632.3952

OpenAI. (2025). *Chatgpt (april 2025 free version) [large language model]*. Retrieved April 1, 2025, from https://chatgpt.com

Pareek, P., & Thakkar, A. (2020). A survey on video-based human action recognition: Recent updates, datasets, challenges, and applications. *Artificial Intelligence Review*, *54*(3), 2259–2322. https://doi.org/https://doi.org/10.1007/s10462-020-09904-8

Pourpanah, F., Abdar, M., Luo, Y., Zhou, X., Wang, R., Lim, C. P., Wang, X.-Z., & Wu, Q. M. J. (2022). A review of generalized zero-shot learning methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–20. https://doi.org/10.1109/TPAMI.2022.3191696

Qi, M., Wang, Y., Qin, J., Li, A., Luo, J., & Van Gool, L. (2020). Stagnet: An attentive semantic rnn for group activity and individual action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(2), 549–565. https://doi.org/https://doi.org/DOI10.1109/TCSVT.2019.2894161

Qin, D., Leichner, C., Delakis, M., Fornoni, M., Luo, S., Yang, F., Wang, W., Banbury, C., Ye, C., Akin, B., Aggarwal, V., Zhu, T., Moro, D., & Howard, A. (2024). Mobilenetv4: Universal models for the mobile ecosystem. In A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler & G. Varol (Eds.), *Computer vision – eccv 2024* (pp. 78–96). Springer Nature Switzerland. https://doi.org/https://doi.org/10.48550/arXiv.2404.10518

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. https://doi.org/https://doi.org/10.48550/arXiv.1506.04214

Singh Gill, H., Ibrahim Khalaf, O., Alotaibi, Y., Alghamdi, S., & Alassery, F. (2022). Fruit image classification using deep learning. *Computers, Materials &amp; Continua*, *71*(3), 5135–5150. https://doi.org/10.32604/cmc.2022.022809

Song, L., Yu, G., Yuan, J., & Liu, Z. (2021). Human pose estimation and its application to action recognition: A survey. *Journal of Visual Communication and Image Representation*, *76*, 103055. https://doi.org/10.1016/j.jvcir.2021.103055

Soomro, K., & Zamir, A. R. (2014). Action recognition in realistic sports videos. In *Computer vision in sports* (pp. 181–208). Springer International Publishing. https://doi.org/10.1007/978-3-319-09396-3_9

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. https://doi.org/https://doi.org/10.48550/arXiv.1409.4842

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., & Paluri, M. (2017). A closer look at spatiotemporal convolutions for action recognition. https://doi.org/https://doi.org/10.48550/arXiv.1711.11248

Wang, H., Jiang, X., Ren, H., Hu, Y., & Bai, S. (2021). Swiftnet: Real-time video object segmentation. https://doi.org/https://doi.org/10.48550/arXiv.2102.04604

Wang, L., Huang, B., Zhao, Z., Tong, Z., He, Y., Wang, Y., Wang, Y., & Qiao, Y. (2023). Videomae v2: Scaling video masked autoencoders with dual masking. https://doi.org/https://doi.org/10.48550/arXiv.2303.16727

Wang, Y., Zhang, J., Zhu, H., Long, M., Wang, J., & Yu, P. S. (2019). Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. https://doi.org/10.1109/CVPR.2019.00937

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2.

Yang, M., Gao, H., Guo, P., & Wang, L. (2023). Adapting short-term transformers for action detection in untrimmed videos. https://doi.org/https://doi.org/10.48550/arXiv.2312.01897

Yin, H., Sinnott, R. O., & Jayaputera, G. T. (2024). A survey of video-based human action recognition in team sports. *Artificial Intelligence Review*, *57*(11). https://doi.org/10.1007/s10462-024-10934-9

Zahan, S., Hassan, G. M., & Mian, A. (2023). Learning sparse temporal video mapping for action quality assessment in floor gymnastics. https://doi.org/10.48550/ARXIV.2301.06103

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., & Lee, B. (2021). A survey of modern deep learning based object detection models. https://doi.org/10.48550/ARXIV.2104.11892

Zhengxia, Z., Keyan, C., Zhenwei, S., Yuhong, G., & Jieping, Y. (2019). Object detection in 20 years: A survey. https://doi.org/https://doi.org/10.48550/arXiv.1905.05055

Zhou, J., Li, H., Lin, K.-Y., & Liang, J. (2023). Towards weakly supervised end-to-end learning for long-video action recognition. https://doi.org/https://doi.org/10.48550/arXiv.2311.17118