Voorspellen van reps in reserve (RIR) bij het uitvoeren van weerstandstraining aan de hand van AI.

Ruben Mattheus.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Dhr. P. Van Der Helst
Co-promotor: Dhr. T. Van Goethem
Academiejaar: 2024–2025
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Als student toegepaste informatica met keuzeriching AI & data engineer was deze bachelorproef een perfecte kans om de technieken gezien tijdens de lessen toe te passen in een groter en persoonlijker project. Ik heb zelf in het verleden bij het raadplegen van een fitness-schema RIR vermeld zien staan, en ik was niet zeker of ik deze aanduiding van intensiteit goed aan het volgen was. De eigen ervaring met het probleem waaraan dit onderzoek een oplossing kan bieden maakte dat ik ook echt interesse heb voor de gevonden resultaten. Het maken van deze bachelorproef ging gepaard met verschillende problemen die zorgden voor interessante conversaties met familie en vrienden, en voor de hulp en inzichten die voortvloeiden uit die gesprekken wil ik hen graag bedanken. Verder wil ik ook mijn promotor, Pieter Van Der Helst, en mijn co-promotor, Thibaut Van Goethem, extra bedanken voor de feedback tijdens het maken van deze bachelorproef.

Samenvatting

Deze bachelorproef heeft als doel om het aantal reps in reserve (RIR) van een sporter tijdens een set repetities in weerstandstraining te voorspellen. De doelgroep van dit onderzoek zijn ontwikkelaars die gebruik kunnen maken van dit model in applicaties, voornamelijk ontwikkelaars van fitness applicaties of onderzoekers die het aantal RIR als variabele willen gebruiken in verder onderzoek. Het verwachte resultaat is een model dat het aantal RIR nauwkeurig kan voorspellen bij zowel ervaren als onervaren sporters. De methodologie die hiervoor toegepast zal worden begint met het kiezen van een toepasselijk model dat met verzamelde data getraind zal worden. De data waarmee gewerkt zal worden is het traject van de barbell bij het uitvoeren van barbell bench press sets. Na het trainen van het model zal de nauwkeurigheid hiervan geëvalueerd worden. Indien het model nauwkeurige voorspellingen kan maken kan het geïmplementeerd worden in fitness applicaties om verder een meer betaalbaar alternatief voor personal coaches aan te bieden of een beter gepersonaliseerd revalidatieplan op te stellen na blessures of ongevallen.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van codefragmenten

1

Inleiding

Weerstandstraining is een belangrijk aspect in het opbouwen van spiermassa en kracht. De beschikbaarheid van commerciële sportscholen maakt individueel trainen op eigen gekozen uren een aantrekkelijke optie voor de doorsnee sporter om aan fysieke activiteit te doen doorheen de week. De vrijheid die komt met het kiezen van eigen uren en oefeningen brengt echter ook problemen met zich mee. Het feit dat begeleiding optioneel is zorgt ervoor dat velen zelf onderzoek doen naar optimale workouts, en voor het grootste deel is het volgen van een schema gelijkaardig aan de service die een personal coach zou leveren. Na het initieel bestuderen van een schema zijn de meeste termen vanzelfsprekend: x sets van y reps op oefening z. Het doseren van de inspanning om optimaal te trainen is echter moeilijk in te schatten.

1.1. Probleemstelling

Het doseren van de inspanning bij het uitvoeren van een oefening krijgt de naam 'reps in reserve', afgekort RIR: trainen tot spierfalen betekent dat er 0 RIR zijn, maar door de vermoeidheid die hieraan vasthangt is het niet aangeraden om dit herhaaldelijk te doen. Wanneer sporters zich hierdoor beginnen in te houden om de optimale 1-2 RIR te houden, schatten ze dit vaak verkeerd in. Een mogelijke oplossing zou een applicatie zijn die aan de hand van videobeelden een voorspelling doet van de hoeveelheid RIR na een set. De sporter krijgt zo feedback die het mogelijk maakt om het trainen verder te optimaliseren.

1.2. Onderzoeksvraag

Met welke nauwkeurigheid kan een applicatie in realtime de RIR van een sporter inschatten? Er zullen verschillende factoren de nauwkeurigheid van deze voorspellingen beïnvloeden. Zo kan een verschil in bijvoorbeeld leeftijd, geslacht of lengte

invloed hebben op de manier waarop de oefeningen uitgevoerd worden. Op welke manier moet de data geclassificeerd worden om de nauwkeurigheid te maximaliseren? De techniek bij sporters met weinig ervaring zal ook minder regelmatig zijn dan die van ervaren sporters. Hoeveel impact heeft een verschil in ervaring tussen twee sporters? Als dit verschil te groot is zullen er extra stappen gezet moeten worden om een model te trainen. Een andere factor kan zijn dat de verzamelde data niet correct geclassificeerd is. Zoals eerder vermeld zijn sporters slecht in het inschatten van hun RIR. Wanneer we het model trainen moeten we zeker zijn dat de sporters hun RIR correct hebben ingeschat. Hoe kunnen we fouten met de data minimaliseren?

1.3. Onderzoeksdoelstelling

Deze bachelorproef dient als proof-of-concept in het voorspellen van de RIR in real-time en het onderzoeken van de beïnvloedende factoren bij deze voorspellingen. Een succesvolle conclusie van deze bachelorproef houdt in dat de voorspellingen accuraat zijn, of dat er te veel factoren zijn om een gewenste nauwkeurigheid te behalen.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:
In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.
In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.
In Hoofdstuk 4 wordt het verwerken van de videobeelden en het berekenen van extra features uitgewerkt, dit is de belangrijkste stap in het genereren van nauwkeurige voorspellingen.
In Hoofdstuk 5 worden de verschillende modellen getraind, het gaat hier om simpele regressiemodellen maar ook om deep learning.
In Hoofdstuk 6 wordt de code toegelicht waarmee in real-time voorspellingen gemaakt kunnen worden, mogelijke verbeteringen en voorstellen voor de implementatie van deze code in een applicatie worden hier ook beschreven.
In Hoofdstuk 7 worden de resultaten van de verschillende modellen vergeleken.
In Hoofdstuk 8, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

In dit hoofdstuk wordt de huidige kennis in het onderzoeksveld toegelicht. Eerst is er een korte toelichting van weerstandstraining, daarna is er meer uitleg over het technische aspect van dit onderzoek.

2.1. Barbell bench press

Om voorspellingen te maken met betrekking tot de barbell bench press is het belangrijk om de oefening voldoende te begrijpen

2.1.1. Opbouw van de oefening

Het uitvoeren van de barbell bench press oefening kan opgesplitst worden in drie delen:

- 'Unracken' van de barbell
- Het excentrische deel
- Het concentrische deel

Het 'unracken' van de barbell is het verplaatsen van de barbell uit het rek waarin de barbell in rust lag. Hierna laat de sporter de barbell zakken tot zijn of haar borst, dit is het excentrische deel van de oefening. Wanneer de sporter de gewenste diepte heeft bereikt, volgt het concentrische deel van de oefening waarbij de sporter de barbell weer naar boven duwt. Spieren in de borst zijn bij de meeste sporters sterker in het excentrische deel van de barbell bench press oefening, waardoor spierfalen vrijwel altijd voorvalt in het concentrische deel.

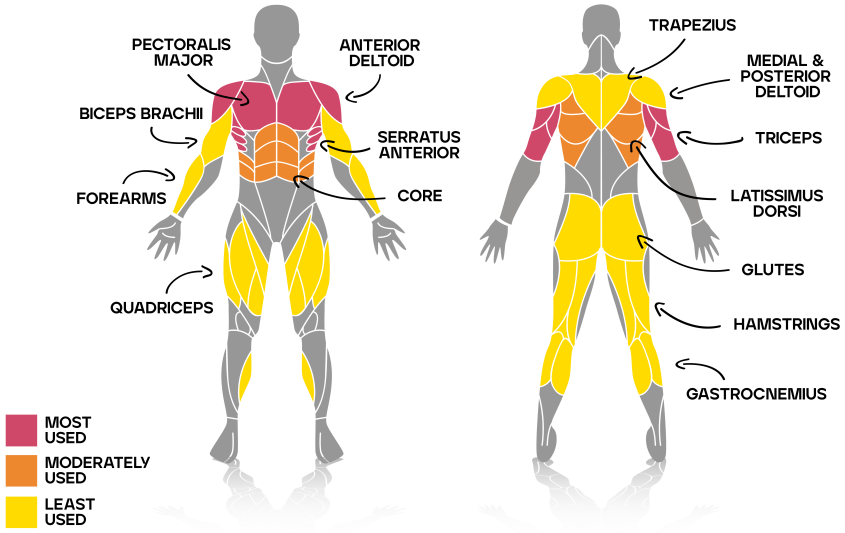
2.1.2. Gebruikte spiergroepen

Verticale beweging van de barbell

Het horizontaal adducteren van de schouders gebeurt door het samentrekken van de 'pectoralis major', de voorste deltoïde spieren en de 'triceps brachii'. Doordat de kracht die tegenwerkt bij het uitvoeren van de barbell bench press de zwaartekracht is, is het verticaal bewegen van de barbell het zwaarste deel van de oefening. De sporter ligt horizontaal, hierdoor is het horizontaal adducteren van de schouders de beweging die zorgt voor de verticale beweging van de barbell. Wanneer de barbell zich op het laagste punt bevindt, wordt vooral de 'pectoralis major' gebruikt. Naarmate de oefening verder uitgevoerd wordt, spelen de voorste deltoïde spieren en de triceps een grotere rol. De breedte waarmee de sporter de barbell vasthoudt heeft ook impact op de spieren die gebruikt worden: meer afstand tussen de handen zorgt voor meer activatie van de 'pectoralis major', minder afstand gaat gepaard met meer stress op de 'triceps'.

Stabilisatie

Naast de spieren die gebruikt worden voor de verticale beweging van de barbell, worden er ook spieren gebruikt voor stabilisatie. De rol van deze spieren is het voorkomen van ongewilde horizontale beweging van de barbell en het bewegen van de sporter op de bank waarop deze zich bevindt. De spieren die zorgen voor de horizontale beweging van de barbell zijn de scapulaire stabilisatoren ('serratus anterior' en de middelste en inferieure 'trapezius') en de rotator cuff-spieren rondom het schoudergewricht. Het voorkomen van de verplaatsing van de sporter zijn de 'transverse abdominis', de 'obliques', 'multifidus', 'erector spinae' en 'quadratus lumborum'.



Figuur 2.1: Grafische voorstelling van de spiergroepen die gebruikt worden bij de barbell bench press oefening.

De spieren aangeduid op deze afbeelding maar niet vermeld in de vorige secties spelen een verwaarloosbare rol.

2.1.3. Voorspellen van de 'one rep max'

Bij het uitvoeren van krachttraining is er sprake van het maximaal gewicht waarmee een individu een bepaalde oefening voor juist 1 repetitie kan uitvoeren, de 'one rep max' of afgekort 1RM. Deze enkele repetitie kan gezien worden als een set tot spierfalen waarbij er maar 1 repetitie is uitgevoerd.
Bij een eerder onderzoek (Babault & Sekulic, 2021) is gevonden dat de snelheid waarmee het uitvoeren van sets van maar 1 repetitie met minder gewicht gebruikt kan worden om de 1RM te voorspellen. Uit de resultaten blijkt dat het voorspellen de hoogste nauwkeurigheid en de laagste fouten vertoont wanneer er meervoudige of geregulariseerde regressie gebruikt wordt.

Geregulariseerde regressie

Geregulariseerde regressie is een regressiemodel waarbij de coëfficiënten zo dicht mogelijk bij nul gehouden worden. Geregulariseerde regressie is een manier om overfitting tegen te gaan.

Overfitting en underfitting

Overfitting en underfitting zijn termen die gebruikt worden om het genereren van een analyse die te specifiek is voor een bepaalde dataset, waardoor voorspellingen op nieuwe data niet succesvol uitgevoerd kunnen worden, te beschrijven. Dit is het gevolg van het model dat naast de eigenlijke eigenschappen ook de noise,

outliers en fluctuaties bestudeert. Overfitting is gekarakteriseerd door een hoge nauwkeurigheid op de trainingsdata, maar een lage nauwkeurigheid op testdata. Het tegenovergestelde hiervan is underfitting.

Noise, outliers en fluctuaties

Noise is een variatie in de data die onafhankelijk van de input-data plaatsvindt, bijvoorbeeld meetfouten. Bij overfitting ziet het model deze noise als input en zullen er dus foute voorspellingen gemaakt worden.
Outliers zijn datapunten die veel verschillen van de andere datapunten. Ook hierbij kan er overfitting plaatsvinden wanneer het model niet de algemene trend volgt maar ook voor alle outliers een correcte waarde wilt voorspellen.
Fluctuaties zijn kleine trends in data die veroorzaakt worden door gebeurtenissen die irrelevant zijn voor het model, bijvoorbeeld nieuws dat ervoor zorgt dat aandeelprijzen plots stijgen. Deze plotse veranderingen in de data kunnen niet voorspeld worden door het model en kunnen overfitting veroorzaken wanneer dit toch geprobeerd wordt.

2.1.4. Foutmarge bij het inschatten van de RIR

De meeste aspecten bij het uitvoeren van weerstandstraining kunnen zeer exact bepaald worden: het gewicht, de hoeveelheid sets, de hoeveelheid reps en welke oefening er juist uitgevoerd wordt zijn objectieve feiten. Een relevant aspect echter is de intensiteit waarmee de set uitgevoerd is. Hiervoor worden regelmatig 2 verschillende termen gebruikt: 'RIR' (Reps In Reserve) of 'RPE' (Rate of Perceived Exertion). De RPE schaal is een schaal van 1-10 waarbij de intensiteit waarmee de oefening uitgevoerd is beoordeeld wordt. Omdat RIR de meer objectieve schaal is zijn er weinig studies waarbij voor RPE gekozen wordt. RIR is echter nog steeds moeilijk in te schatten: een relevante studie over de nauwkeurigheid waarmee getrainde proefpersonen de RIR inschatten toont dat zelfs bij getrainde personen er een foutmarge van 1 repetitie zit (Refalo, 2023).

2.2. Object detection models

Er zijn veel verschillende models die gebruikt kunnen worden om een barbell te tracen, deze modellen kunnen geclassificeerd worden als 'Single-stage detectors' of 'Two-stage detectors'. Om dit onderzoek succesvol uit te voeren zal er een object detection model gebruikt moeten worden om de positie van de barbell in de video vast te kunnen leggen. In de komende subsecties worden de verschillende mogelijkheden kort toegelicht.

2.2.1. Two-stage detectors

(voorbeelden: Faster R-CNN, Mask R-CNN, Detectron2)
De eerste stage bij een Two-stage detector is het genereren van 'Region propo-

sals' waarbij een 'Region Proposal Network', afgekort RPN, een poging doet om de afbeelding op te splitsen in verschillende regio's. Hierna wordt er classificatie toegepast op de individuele regio's. Deze aanpak zorgt voor hoge nauwkeurigheid bij het voorspellen van de objecten in een afbeelding.

2.2.2. Single-stage detectors

(voorbeelden: YOLO, SSD, RetinaNet)
Bij Single-stage detectors worden de eerste en tweede stage van Two-stage detectors simultaan uitgevoerd, dit wil zeggen dat wanneer een regio geclassificeerd is er ook meteen een bounding box en class score aan toegevoegd worden. Dit heeft als gevolg dat Single-stage detectors sneller maar minder nauwkeurig zijn dan Two-stage detectors.

2.3. Pose estimation

Een andere optie om de positie van een barbell te tracen is het gebruik van 'pose estimation'. Bij pose estimation worden belangrijke punten gedetecteerd en getrackt. Bij mensen zijn dit bijvoorbeeld handen en voeten, knieën, schouders; bij objecten hangt dit af van object tot object: bij auto's worden bijvoorbeeld vaak de lichten gebruikt bij pose estimation.
Bij pose estimation zijn er twee verschillende groepen van methodes om de 'pose' te detecteren: enerzijds zijn er de bottom-up methodes waarbij de belangrijke punten eerst gedetecteerd en daarna samengevoegd worden om de 'pose' te vormen, anderzijds zijn er de top-down methodes waarbij eerst het object en dan de kernpunten gedetecteerd worden.

2.3.1. Bottom-up of top-down pose estimation?

In het algemeen zijn bottom-up methodes sneller dan top-down methodes omdat top-down methodes eerst object detection uitvoeren, waarna binnen het gedetecteerde object de kernpunten gezocht worden. Top-down methodes hebben als voordeel dat ze nauwkeuriger zijn en dat er minder problemen zijn wanneer er meerdere objecten in beeld zijn.

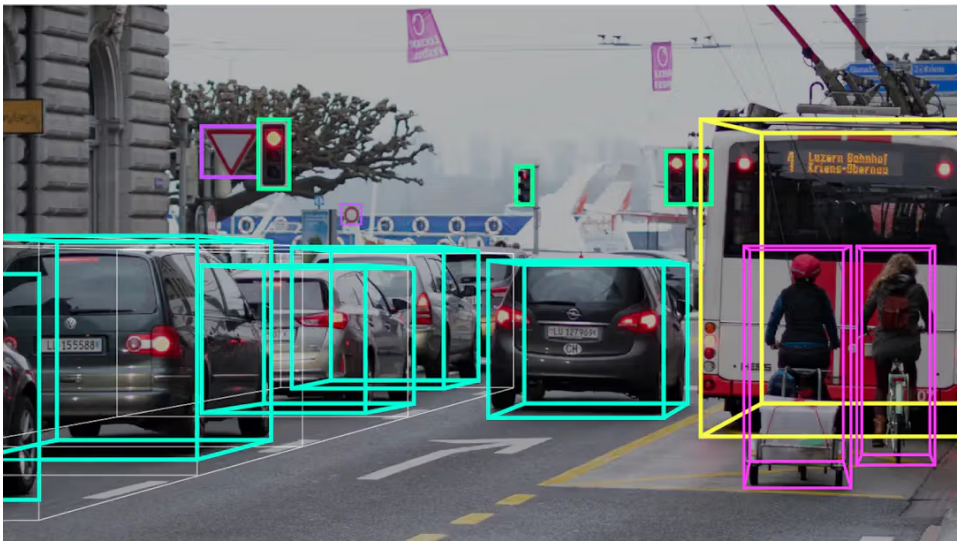
2.3.2. Pose estimation of object detection models?

Het voordeel van pose estimation is dat naast het herkennen van objecten ook met de oriëntatie rekening gehouden wordt. Dit zorgt voor meer informatie bij het detecteren van objecten maar is ook bijgevolg minder efficiënt.

2.4. Bounding boxes

'Bounding boxes' zijn rechthoekige begrenzingen van objecten die gebruikt worden in 'computer vision' om de locatie van de gedetecteerde objecten te definiëren.

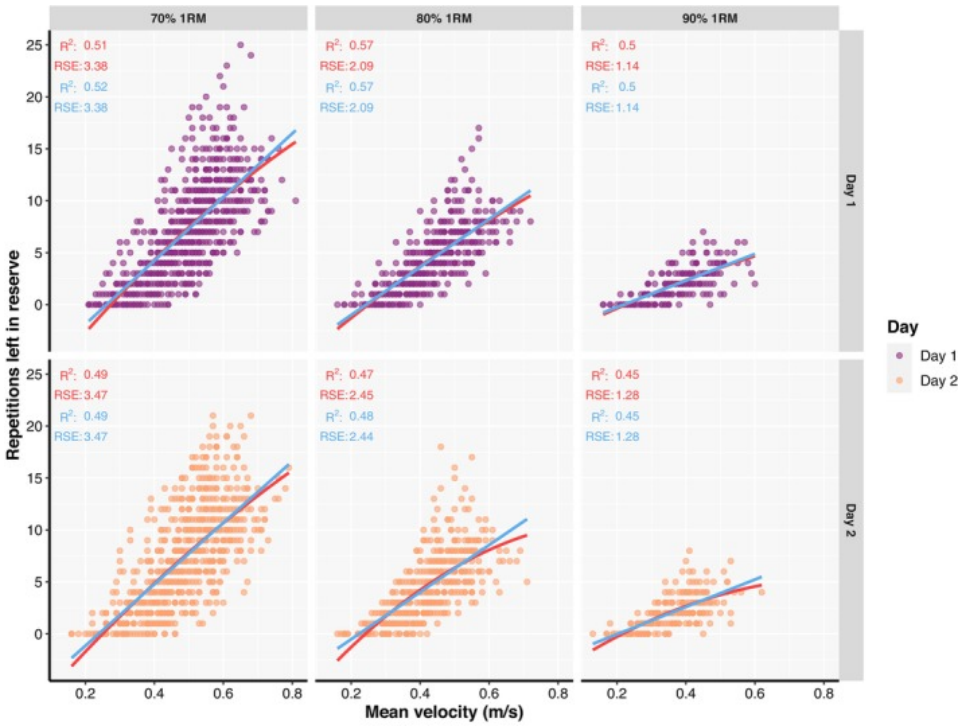
Hiervoor worden er 4 coördinaten gebruikt: x_min, x_max, y_min en y_max. Deze coördinaten kunnen opgevraagd worden wanneer er een video verwerkt wordt met computer vision.



Figuur 2.2: Bounding boxes in computer vision.

2.5. Model

Het voorspellen van de RIR gebaseerd op de snelheid van de eerdere reps in de set is eerder succesvol uitgevoerd in een labo waarbij duidelijk werd dat dit verband voorgesteld kon worden met regressie (Jukic e.a., 2024). In dit onderzoek werd er gewerkt met de 'back squat' oefening.
Uit deze studie blijkt dat het verband tussen de snelheid van repetities en de RIR niet heel nauwkeurig kan voorgesteld worden voor een groep mensen, en dat het onderzoeken van een individueel verband veel betere resultaten oplevert. Ook is in deze studie duidelijk geworden dat bij vrouwen, zwakkere individuen en mensen met minder dan 3 jaar ervaring in weerstandstraining de RIR minder nauwkeurig voorspeld kan worden. Daarnaast is het verband tussen de snelheid van repetities en de RIR ook afhankelijk van het gewicht waarmee de oefening uitgevoerd wordt. Het eigenlijke gewicht is niet vereist, eerder het verband tussen de 1RM en het gewicht (bijvoorbeeld 60% van de 1RM).



Figuur 2.3: Het verband tussen RIR en gemiddelde snelheid bij het uitvoeren van de set. De rode lijn stelt lineaire regressie voor, de blauwe lijn is tweede orde polynomiale regressie.

3

Methodologie

3.1. Verzamelen van data

Om te beginnen zal er data verzameld worden in de vorm van videobeelden van proefpersonen met variërende ervaring met weerstandstraining waarbij een set tot spierfalen uitgevoerd wordt. Het feit dat de video's eindigen met spierfalen betekent dat er een constante 0 RIR is bij de beelden.
De data zal bestaan uit sets van verschillende vrijwilligers, gefilmd in een commerciële sportschool. Bij enkele proefpersonen zal extra data verzameld worden zodat de voor- en nadelen van een uniek model per persoon onderzocht kunnen worden.

3.2. Verwerken van data

De data waarmee het model zal werken is het resultaat van het verwerken van de videobeelden waarbij het traject van de barbell onderzocht wordt aan de hand van computer vision. Er moet een methode uitgewerkt worden om in real-time de snelheid van de barbell tijdens de repetitie te bepalen.

3.3. Opstellen van het model

Nadat de data verwerkt is, kan een model getraind worden op de nuttige data uit de video's. Dit model zal op basis van de snelheid van de repetities in een set een voorspelling doen over de RIR.

3.4. Onderzoeken van de invloed van verschillende factoren

Bij het opstellen van het model zal onderzocht worden welke factoren de meeste invloed hebben op de nauwkeurigheid van het model. Indien blijkt dat sommige

factoren geen of verwaarloosbare invloed hebben kan het model veralgemeend worden. Blijkt echter dat sommige factoren veel invloed hebben zullen er om de nauwkeurigheid te maximaliseren verschillende modellen opgesteld moeten worden. De voornaamste factoren die onderzocht zullen worden zijn de verschillen van persoon tot persoon, ervaring van de sporter en of vermoeidheid voor het beginnen van de sets invloed heeft op het verband tussen snelheid van de repetitie en RIR.

3.5. Testen en evalueren

Wanneer het model getraind is kunnen er nieuwe, ongeziene beelden gebruikt worden om het model te testen en kan de nauwkeurigheid van het model geëvalueerd worden.

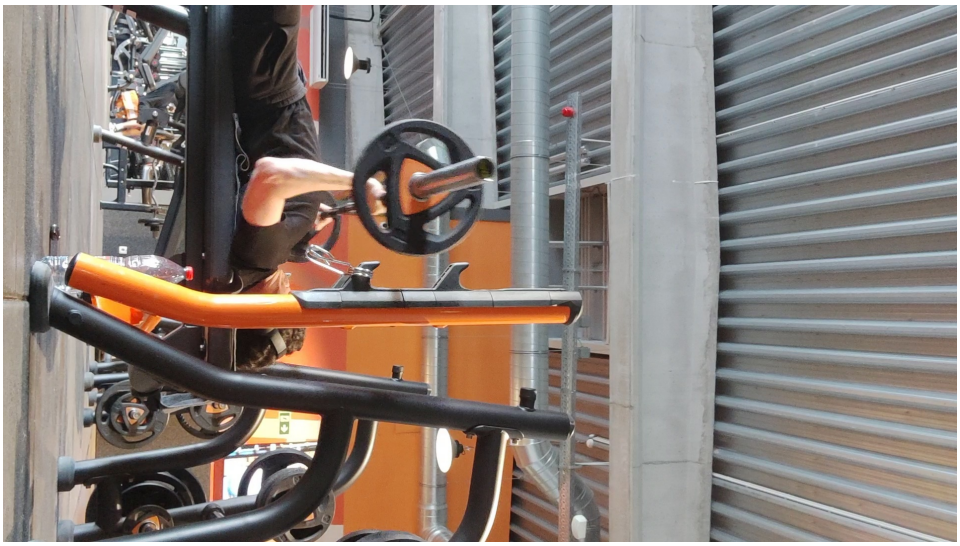
4

Data

Dit is het startpunt van het onderzoek: het verzamelen van data in een formaat waar uiteindelijk verder mee gewerkt kan worden.

4.1. Videobeelden

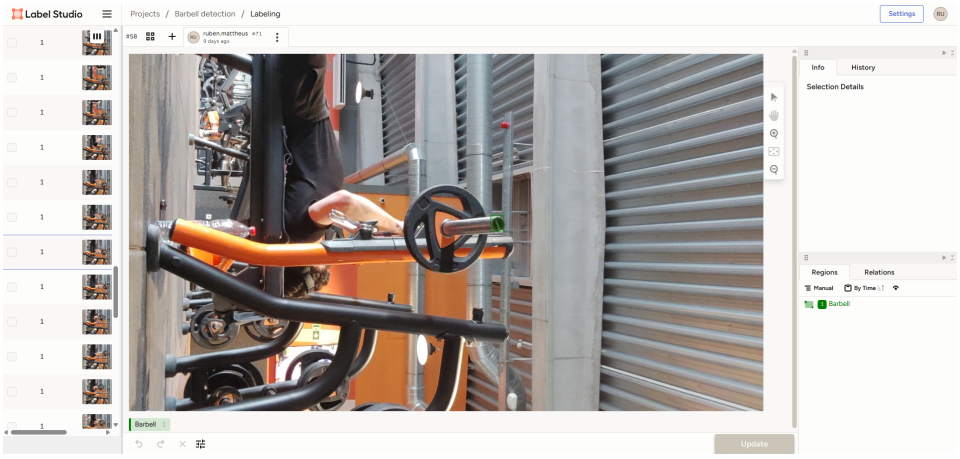
Dit onderzoek dient als proof of concept en de focus ligt dus niet op het maken van een applicatie die werkt in elke situatie, alle video's die dienen als data worden gefilmd op dezelfde locatie in gelijkaardige omstandigheden om zo weinig mogelijk ruis te hebben bij de data.



Figuur 4.1: Frame 1600 in een video van ongeveer 2500 frames.

4.2. Label-Studio

Label-Studio is een labeling platform dat kan dienen als labeling voor een AI object detection model, in dit geval worden er eerst op 100 frames bounding boxes getekend op het uiteinde van een barbell. Zolang het uiteinde van de barbell uniek en optioneel iets van tekst of afbeelding bevat, is de nauwkeurigheid hoog genoeg zodat dit voor accurate data kan zorgen.



Figuur 4.2: Het labelen van de data in label-studio.

Het geëxporteerd model kan getraind worden en dienen als YOLO model.

4.3. Ruwe data

Met behulp van een Python script kan het geëxporteerde model gebruikt worden om de coördinaten van de barbell op te slaan in een .csv bestand: in elke frame van de input worden barbells gedetecteerd door het model, het middelpunt van de bounding box wordt als coördinaat opgeslagen. De frames worden ingelezen aan de hand van de OpenCV library.

```
    for box in results.boxes:
        cls = int(box.cls[0])
        if cls == 0:
            barbell_in_frame = True
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            x_center = (x1 + x2) // 2
            x_coords.append((frame_nr, x_center))
            break

    if not barbell_in_frame:
        x_coords.append((frame_nr,))
```

Codefragment 4.1: Verwerken van de data in een frame van een video.

Deze code veronderstelt dat de videobeelden verticaal gefilmd zijn met een gsm, OpenCV draait deze echter 90 graden waardoor de hoogte van de barbell voorgesteld is met het x-coördinaat. Er wordt enkel met de hoogte van de barbell gewerkt omdat een horizontale beweging bij het uitvoeren van een bench press repetitie veroorzaakt wordt door spieren die geen rechtstreekse invloed hebben op het bereiken van spierfalen in een set. Wanneer er geen barbell gedetecteerd is in een bepaalde frame, wordt de frame zonder coördinaat opgeslagen, deze ontbrekende datapunten worden later aangevuld.

4.4. Start en stop van repetities

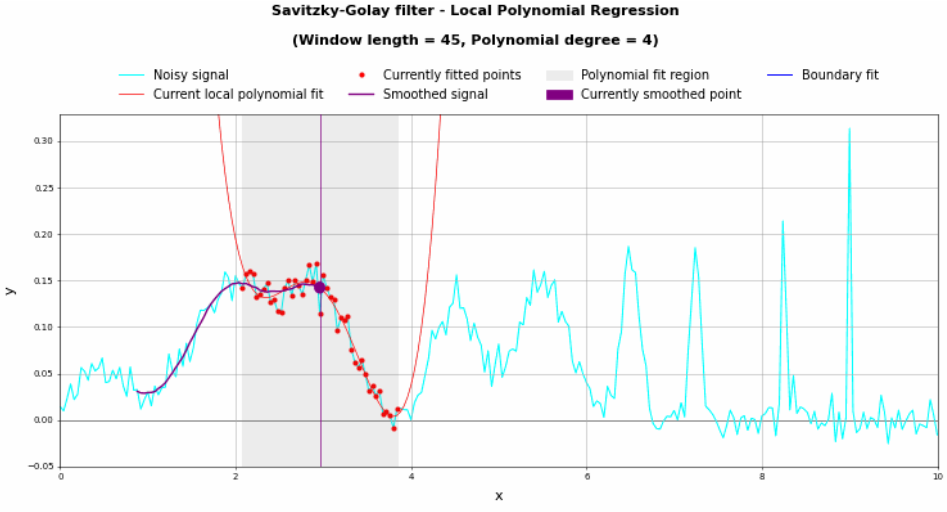
De ruwe data in de .csv bestanden zijn niet genoeg om een model op te trainen, er zit weinig informatie in enkel coördinaten. Het is bijvoorbeeld ook belangrijk om aan te duiden waar een repetitie start en stopt. Hiervoor maken we eerst enkele veranderingen aan de ruwe data:

```
1 df['Coordinate'] = df['Coordinate'].interpolate(method='linear')
2
3 df['savgol_coordinate'] = savgol_filter(df['Coordinate'],
      window_length=15, polyorder=2)
4 df['savgol_velocity'] = np.gradient(df['savgol_coordinate'])
```

Codefragment 4.2: Aanpassingen aan de ruwe data.

Het interpoleren van de coördinaten is belangrijk voor frames waarin de barbell niet herkend werd. De oorzaak hiervan kan verschillen, maar aangezien er bij bench press repetities geen onverwachte bewegingen gebeuren zorgt een simpele interpolatie voor het opvullen van de ontbrekende datapunten. Kleine fouten in de detectie van de barbell zorgen al snel voor moeilijkheden in het werken met de data, het toepassen van de Savitzky-Golay filter zorgt ervoor dat kleine bewegingen niet fout geïnterpreteerd worden.

Het start- en stoppunt van een repetitie is makkelijker aan te duiden met de nieuwe data aangezien een kleine beweging in de barbell niet meer te zien is in de data. Dit maakt het mogelijk om een tekenverschil in de snelheid als indicator te gebruiken. Om het start- en stoppunt aan te duiden wordt er rekening gehouden met de afstand die de barbell heeft afgelegd. Hiervoor worden er variabelen gedefiniëerd om het laagste en hoogste punt van de barbell in de video op te slagen. Ook voor de positie van de barbell in de eerste frame is een variabele gedefiniëerd. In de dataframe waarin de coördinaten opgeslagen zijn wordt ook een kolom voor de 'rep state' toegevoegd, deze zal dienen om als waarde 'start' of 'stop' te vertonen. Bij het uitoefenen van de oefening zal eerst de barbell lichtjes omhoog gaan bij het 'unracken' van de barbell, opdat dit niet gezien zal worden als een repetitie is de



Figuur 4.3: Visualisatie van de Savitzky-Golay filter die toegepast wordt op een grafiek.

'in_rep' variabele toegevoegd. Deze variabele dient True te zijn van het begin van de repetitie tot het einde ervan, en zal False zijn wanneer de barbell van het einde van de vorige repetitie tot de start van de volgende repetitie verplaatst wordt.

```
for i in range(1, len(df)):
    current_velocity = df.iloc[i]['savgol_velocity']
    coordinate = df.iloc[i]['savgol_coordinate']

    if top is None and bottom is None:
        if np.isnan(coordinate):
            continue
        top = coordinate
        bottom = coordinate
        start_coordinate = coordinate
```

Codefragment 4.3: Startwaardes in verband met de 'range of motion' opslaan.

De waardes in verband met de 'range of motion' worden ingesteld op de initiële coördinaat van de barbell. Deze variabelen zijn handig bij het detecteren van een repetitie en moeten doorheen de set bijgewerkt worden.

```
if coordinate > top:
    top = coordinate
    range_of_motion = top - bottom

elif coordinate < bottom:
    bottom = coordinate
    range_of_motion = top - bottom
```

Codefragment 4.4: Updaten van de 'range of motion'.
Bij elke frame van de data wordt gecontroleerd of er maximum of minimum in de coördinaten bereikt is. Deze waardes zorgen ervoor dat wanneer een repetitie niet succesvol uitgevoerd kan worden, deze verandering in richting niet als het stoppen van een repetitie gezien wordt.

```
if not in_rep and current_velocity > 0:
    counter += 1
elif not in_rep and current_velocity < 0:
    counter = 0
if in_rep and current_velocity < 0:
    counter += 1
elif in_rep and current_velocity > 0:
    counter = 0
```

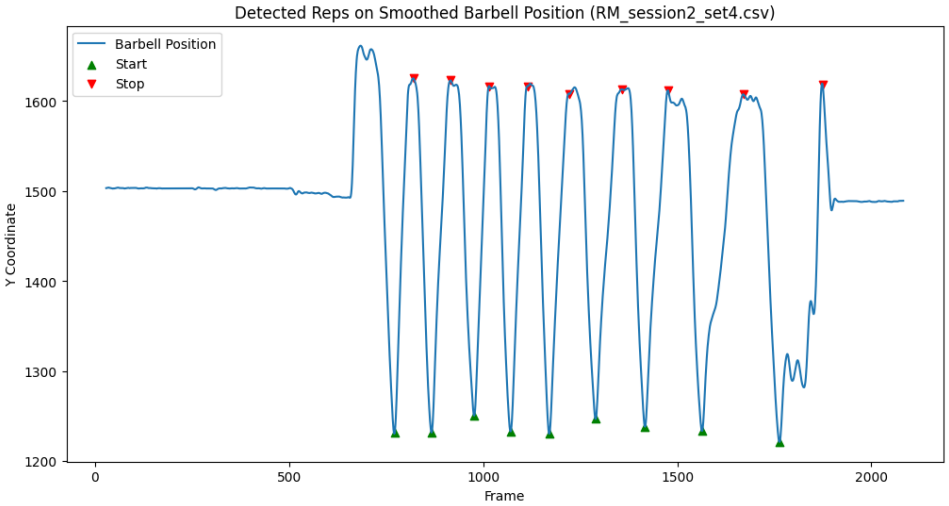
Codefragment 4.5: Counter updaten bij het bewegen van de barbell.
Wanneer de barbell de verwachte verplaatsing vertoont (omhoog tijdens een repetitie, omlaag tussen twee repetities) reset de counter naar 0. Beweegt de barbell echter in de tegengestelde richting, dan gaat de counter met 1 omhoog.

```
if not in_rep and counter ⩾ counter_threshold and coordinate <
    bottom + range_of_motion * 0.25 and abs(coordinate -
    start_coordinate) > 50:
    in_rep = True
    df.at[i, 'rep_state'] = "start"

elif in_rep and counter ⩾ counter_threshold and coordinate > top -
    range_of_motion * 0.25:
    df.at[i, 'rep_state'] = "stop"
    in_rep = False
```

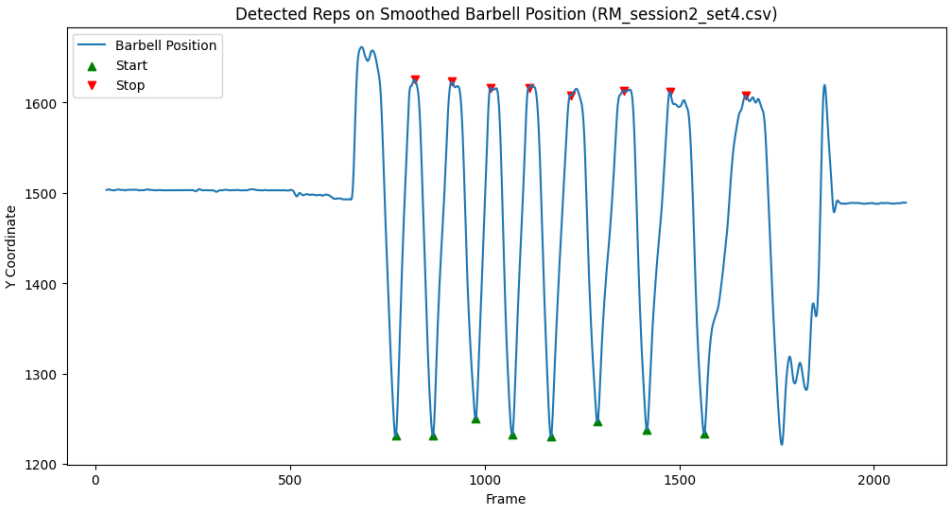
Codefragment 4.6: Toevoegen van start- en stoppunten in de dataframe.
Wanneer de counter een waarde gelijk of hoger dan de counter_threshold heeft, wordt dit gezien als een start- of stoppunt van een repetitie. Een start van een repetitie kan enkel plaatsvinden wanneer de barbell zich in de onderste 25% van de 'range of motion' bevindt, en een repetitie kan enkel stoppen wanneer de barbell in de bovenste 25% is. Deze code veronderstelt dat de barbell op de beginpositie ten minste 50 pixels boven het laagste punt start zodat het 'unracken' van de barbell

niet als een eerste repetitie gezien wordt. De counter_threshold zou dienen als hulpmiddel zodat een korte verandering in de richting waarin de barbell beweegt niet als begin of einde van een repetitie zou tellen, de Savitzky-Golay filter maakt dit echter overbodig en een waarde van 1 voor de counter_threshold heeft nog steeds het gewenste resultaat.



Figuur 4.4: Visualisatie van het aanduiden van het begin en einde van een repetitie.

De videobeelden waaruit de data opgemaakt zijn, zijn steeds van een set tot spierfalen, hierdoor valt op de vorige afbeelding te zien dat de laatste repetitie eerst even omhoog gaat, dan weer zakt om dan plots weer omhoog te schieten. Dit is omdat door vermoeidheid de laatste repetitie niet zelfstandig vervolledigd kon worden en dat er hulp nodig was. Deze laatste repetitie is dus geen deel meer van de data en moet niet aangeduid worden met een start- en stoppunt. De laatste start- en stoppunten worden verwijderd, wanneer de laatste stop echter niet aangeduid is wordt enkel de start verwijderd. De manier waarop de start- en stoppunten aangeduid worden maakt het onmogelijk dat er enkel een stoppunt is.



Figuur 4.5: Op de grafiek waarop de repetities aangeduid zijn valt te zien dat de laatste repetitie (waarbij geholpen is) verwijderd kan worden.

4.5. Extra features

Na het aanduiden van het begin en einde van elke geldige repetitie is het mogelijk om de eigenlijke eigenschappen van deze repetities te gaan bekijken.

```
for i in range(1, len(df)):
    if df.iloc[i]['rep_state'] == 'start':
        start_frame = i

    elif df.iloc[i]['rep_state'] == 'stop':
        stop_frame = i
        frame_count = stop_frame - start_frame
        average_velocity = df['savgol_velocity'].iloc[start_frame :
            stop_frame].mean()
        range_of_motion = df.iloc[stop_frame]['savgol_coordinate'] -
            df.iloc[start_frame]['savgol_coordinate']
        reps.append([frame_count, average_velocity, range_of_motion])
```

Codefragment 4.7: Het opslaan van repetitie-data.

Aan de hand van het start- en stoppunt wordt bij elke repetitie de tijd (voorgesteld in hoeveelheid frames) nodig om de repetitie te voltooien, de gemiddelde snelheid van de barbell tijdens de repetitie, en de 'range of motion' bij de repetitie uitgerekend. De tijd is uitgedrukt in frames, dat wil zeggen dat bij toestellen waarbij video's gefilmd worden in een hogere 'framerate' dit model slechter zal presteren, zolang de 'framerate' echter hetzelfde blijft is deze eenheid goed genoeg om een model te trainen. De afstand is uitgedrukt in pixels en de snelheid in pixels/frame.

Deze eenheden zullen zorgen voor een minder nauwkeurig model wanneer niet elke video gefilmd is vanop dezelfde plaats.

```
rep_df = pd.DataFrame(reps, columns=['time', 'average velocity',
    'range of motion'])

rep_df['rep'] = rep_df.index.map(lambda i : i + 1)
rep_df['RIR'] = rep_df.index.map(lambda i : len(rep_df) - i - 1)
```

Codefragment 4.8: Toevoegen van 'rep' en 'RIR' kolom aan de dataframe.

Aan de dataframe waarin de repetities opgeslagen zijn worden een 'rep' kolom en een 'RIR' kolom toegevoegd. De gefilmde set werd uitgevoerd tot spierfalen, de 'RIR' kolom is dan gelijk aan het verschil van de lengte van de volledige set en het aantal repetities die al uitgevoerd zijn.

```
rep_df['normalized velocity'] = rep_df['average velocity'] /
    rep_df['range of motion']
```

Codefragment 4.9: Normaliseren van de gemiddelde snelheid van de repetities.

Het normaliseren van de snelheid wordt uitgevoerd door de gemiddelde snelheid te delen door de 'range of motion', hiermee is de snelheid opgeslagen in een eenheid die niet afhangt van de afstand aangezien de schaal van deze afmeting kan verschillen tussen twee video's.

```
first_rep_velocity = rep_df.iloc[0]['normalized velocity']

rep_df['first rep velocity'] = first_rep_velocity
rep_df['velocity change since first rep'] = rep_df['normalized
    velocity'] - rep_df['first rep velocity']
rep_df['velocity change (%) since first rep'] = 100 *
    (rep_df['normalized velocity'] - rep_df['first rep velocity']) /
    rep_df['first rep velocity']

rep_df['velocity change since last rep'] = rep_df['normalized
    velocity'].diff().fillna(0)
rep_df['velocity change (%) since last rep'] = rep_df['normalized
    velocity'].pct_change() * 100
rep_df['velocity change (%) since last rep'] = rep_df['velocity
    change (%) since last rep'].fillna(0)
```

Codefragment 4.10: Het toevoegen van extra features aan de repetitie-data.

Om vermoeidheid uit te drukken worden enkele extra features uitgerekend en toegevoegd aan de dataframe, in deze berekeningen wordt steeds de genormaliseerde snelheid gebruikt. Om de vermoeidheid doorheen de hele 'set' te bekijken is bij elke rij de snelheid van de eerste repetitie opgeslagen, elke rij heeft dan een kolom met het verschil tussen de snelheid van de huidige repetitie en de eerste repetitie, uitgedrukt in de genormaliseerde snelheid en in een percentage. Op dezelfde manier is het verschil tussen de huidige en de vorige repetitie berekend.

5

Model

Na het verwerken van de data uit de videobeelden is de volgende stap van het onderzoek om te kijken welk model het beste gebruikt kan worden om de RIR te voorspellen.

5.1. Lineaire regressie

Er is een verband tussen de snelheid waarmee de repetitie werd uitgevoerd en de hoeveelheid RIR, ook zijn de features berekend in Hoofdstuk 4 niet complex genoeg om lineaire regressiemodellen niet te bekijken. Het startpunt bij het trainen van de regressiemodellen ziet er steeds gelijkaardig uit:

```
1 X = df[['time', 'average velocity', 'range of motion', 'rep',
      'normalized velocity', 'first rep velocity', 'velocity change
      since first rep',
2         'velocity change (%) since first rep', 'velocity change since
              last rep', 'velocity change (%) since last rep']]
3 y = df['RIR']
4
5 X_train, X_test, y_train, y_test = train_test_split(X, y,
      test_size=0.2, random_state=42)
```

Codefragment 5.1: Startpunt bij het trainen van een regressiemodel, alle .csv bestanden die voldoen aan de expressie worden ingelezen en opgesplitst in training- en testdata.

80% van de data dient als trainingdata, de resterende 20% is testdata. De gemiddelde snelheid die niet genormaliseerd is wordt weggelaten. Bij het trainen van de modellen wordt steeds random_stage 42 gebruikt zodat de resultaten niet verschillen wanneer de code meerdere keren uitgevoerd wordt.

5.1.1. RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Codefragment 5.2: Evalueren en opslaan van een RandomForestRegressor model.

Een RandomForestRegressor model heeft als voordeel dat het niet gevoelig is voor overfitting, dit geeft het model een hogere nauwkeurigheid bij ongeziene data. In het RandomForestRegressor model worden 100 n_estimators gebruikt. Bij lagere waardes is de kans op underfitting groot, bij hogere waardes zal het model trager worden en bij andere modellen vergroot de kans op overfitting. Ook bij de volgende regressiemodellen zullen er 100 n_estimators gebruikt worden.

5.1.2. GradientBoostingRegressor

```
from sklearn.ensemble import GradientBoostingRegressor

model = GradientBoostingRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Codefragment 5.3: Evalueren en opslaan van een GradientBoostingRegressor model.

Een GradientBoostingRegressor model heeft het potentieel om nauwkeuriger te zijn dan een RandomForestRegressor model, maar is gevoeliger voor noise in de data. Omdat overfitting hier een risico is, is het belangrijk dat de data geregulariseerd is.

5.1.3. XGBRegressor

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)

model = XGBRegressor(n_estimators=100, max_depth=4)
model.fit(X_train, y_train)
```

Codefragment 5.4: Evalueren en opslaan van een XGBRegressor model.

Bij een XGBRegressor model hangt het resultaat meer af van de hyperparameters dan bij de vorige modellen: een hogere max_depth parameter geeft een nauwkeuriger resultaat maar is minder generaliseerbaar.

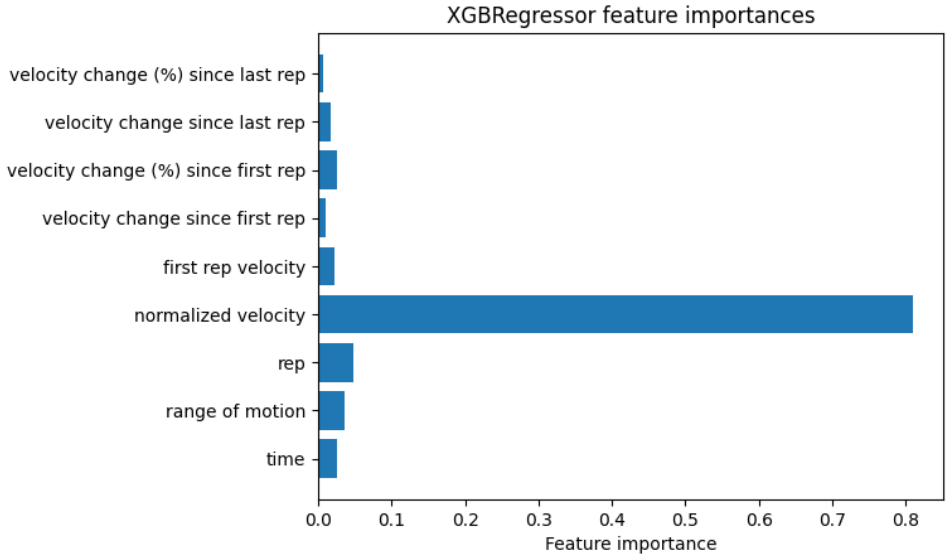
5.1.4. Feature importance

Met de volgende code kan onderzocht worden welke features de grootste rol spelen in het voorspellen van de RIR. Het belang van de features ligt bij de drie regressiemodellen dicht bij elkaar.

```
importances = model.feature_importances_
features = X.columns
```

Codefragment 5.5: Feature importance bij regressiemodellen onderzoeken.

Wanneer deze waarden voorgesteld worden in een grafiek krijgen we het volgende te zien:



Figuur 5.1: Feature importance bij regressiemodellen.

Hieruit blijkt dat de extra features (het vertragen tegenover de eerste en vorige repetitie) weinig invloed hebben op de voorspelling gemaakt door de regressiemodellen. Om het verband tussen de vermoeidheid en de RIR te gebruiken bij het maken van de voorspellingen dienen dus complexere modellen gebruikt te worden.

5.2. Deep learning

Om meer dan enkel de snelheid te gebruiken om voorspellingen te maken moet er met deep learning gewerkt worden, deze modellen kunnen meer complexe verbanden aanleren waardoor extra features zorgen voor een hogere nauwkeurigheid.

5.2.1. CNN

Een Convolutional Neural Network (CNN) werkt met een vaste input shape maar is efficiënt en werkt goed met de eerdere waardes in de data. Dit zorgt dat de data

eerst opgesplitst moet worden om het model te trainen. Het is belangrijk dat deze groeperingen niet gemaakt worden met data van twee verschillende sets. Hiervoor is er een unieke 'set_id' toegevoegd aan de data. Daarna wordt 'X', gebaseerd op deze 'set_id', opgevuld met groepen van 3 repetities.

```
num_samples, num_steps, num_features = X.shape
X_flat = X.reshape(-1, num_features)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_flat)
X = X_scaled.reshape(num_samples, num_steps, num_features)
```

Codefragment 5.6: Feature scaling voor CNN.

Om de features te schalen worden deze eerst omgezet van een 3D array naar een 2D array waarna de features geschaald worden zodat elke feature een gemiddelde van 0 en een standaardafwijking van 1 heeft. Nadat het schalen is uitgevoerd worden de features weer omgezet naar een 3D array.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

Codefragment 5.7: Splitsen van data voor het trainen en testen.

De data wordt gesplitst in training- en testdata. 80% van de data dient om het trainen van het model uit te voeren, de resterende 20% dient om te testen. Om het model te maken wordt er hyperparameter tuning met behulp van een aangepaste 'loss' functie toegepast:

```
def custom_weighted_mae(y_true, y_pred):
    weights = K.exp(-y_true / 5.0)
    return K.mean(weights * K.abs(y_true - y_pred), axis=-1)
```

Codefragment 5.8: Custom 'loss' functie bij het trainen van een CNN.

Deze 'loss' functie zorgt ervoor dat fouten bij lagere RIR waarden zwaarder doorwegen bij het evalueren van het model, wat zal zorgen dat de hyperparameters getuned worden voor een meer accuraat model rond 0-4 RIR. De functie om het CNN op te stellen is de volgende:

```
def build_model(hp):
    model = Sequential()

    model.add(Conv1D(
        filters=hp.Int('conv1_filters', min_value=32, max_value=128,
            step=32),
        kernel_size=hp.Choice('kernel_size', values=[2, 3, 5]),
        activation='relu',
        input_shape=(3, 8),
        padding='same'
    ))

    model.add(BatchNormalization())

    model.add(Conv1D(
        filters=hp.Int('conv2_filters', min_value=32, max_value=128,
            step=32),
        kernel_size=hp.Choice('kernel2_size', values=[2, 3]),
        activation='relu',
        padding='same'
    ))

    model.add(GlobalMaxPooling1D())
    model.add(Dropout(hp.Float('dropout', min_value=0.1,
        max_value=0.5, step=0.1)))

    model.add(Dense(hp.Int('dense_units', min_value=32,
        max_value=128, step=32), activation='relu'))
    model.add(Dense(1))

    model.compile(
        optimizer=Adam(hp.Choice('lr', values=[1e-3, 1e-4])),
        loss=custom_weighted_mae,
        metrics=['mae']
    )

    return model
```

Codefragment 5.9: Functie om een CNN model op te stellen aan de hand van hyperparameter tuning.

Het CNN model heeft twee 'Conv1D' lagen, 'pooling' en 'dropout' voor regularisatie en 'dense' lagen om voorspellingen te maken. Met Keras Tuner worden er verschillende hyperparameters getest om uiteindelijk de parameters die zorgen voor de beste resultaten op te slaan:

```
tuner = RandomSearch(
    build_model,
    objective='val_mae',
    max_trials=20,
    executions_per_trial=1,
    directory='cnn_tuning',
    project_name='cnn_rir'
)

tuner.search(X_train, y_train,
             validation_split=0.2,
             epochs=30,
             batch_size=16,
             callbacks=[EarlyStopping(patience=5)])
```

Codefragment 5.10: Tunen van de hyperparameters.

De 'RandomSearch' functie probeert verschillende hyperparameter combinaties uit en kiest de beste, gebaseerd op de 'objective' parameter (hier ingesteld op 'val_mae'). Er worden maximaal 20 combinaties geprobeerd en bij elke combinatie wordt het model 1 keer getraind. Met 'tuner.search' worden de eigenlijke hyperparameter configuraties getraind aan de hand van de trainingsdata.

```
best_model = tuner.get_best_models(num_models=1)[0]

best_model.fit(X_train, y_train, epochs=10, batch_size=16)
```

Codefragment 5.11: Opnieuw trainen van het model met de gevonden hyperparameters.

Hier wordt het model met de hoogste nauwkeurigheid opnieuw getraind op de volledige dataset aangezien het de eerste keer getraind was op enkel de testdata, waarna het model klaar is om voorspellingen te maken.

6

Live-data

Er is een manier om de ruwe data om te zetten naar bruikbare gegevens en er is een model getraind op deze gegevens, het resterende technische aspect van dit onderzoek is het implementeren van beide in een script dat live videobeelden analyseert.

6.1. Python script met OpenCV

Om voorspellingen te maken op videobeelden in real-time kunnen afzonderlijke stukken code uit de vorige hoofdstukken samengevoegd worden. Deze aanpak heeft voor een groot deel het gewenste resultaat, het grootste probleem zijn de berekeningen die uitgevoerd moeten worden op de data: het toevoegen van nieuwe data aan de dataframe aan een hoog tempo (60 keer per seconde), het interpoleren van de data en het toepassen van de Savitzky-Golay filter zijn niet toepasselijk op mobiele apparaten in real-time. Een verbetering is dat de dataframes niet elke frame aangevuld worden, in plaats daarvan wordt er een buffer dictionary toegevoegd waarin 15 datapunten opgeslagen worden, elke 15 frames worden deze dan toegevoegd aan de dataframe waarna gecontroleerd wordt of er in de laatste 15 frames een repetitie gestart of gestopt is.

6.2. Object detection model

YOLO is een inefficiënt model om mee te werken, om het detecteren van de barbell minder intensief te maken is een 'nano model' zoals YOLOv5n of YOLOv8n een betere optie. Ook kan het detecteren van de barbell elke X aantal frames (bijvoorbeeld elke 3 frames) gebeuren in plaats van elke frame. Na het interpoleren van de barbell positie in de overige 2 frames zou het resultaat gelijkaardig zijn. Het detecteren van een object vraagt ook meer werk dan het detecteren van een 'keypoint' in een object, 'keypoint detection' zou dus ook een mogelijke verbetering kunnen

zijn.

6.3. NumPy

Het uitvoeren van berekeningen op een pandas dataframe is te traag voor real-time applicaties, de code zou herschreven kunnen worden om NumPy te gebruiken ter vervanging van de pandas dataframe waardoor het toevoegen van de frame-coördinaat data efficiënter verloopt. NumPy maakt gebruik van geoptimaliseerde C-backends, de NumPy-arrays zijn in staat om nieuwe waarden aan het einde van de bestaande data toe te voegen. Om nieuwe data toe te voegen in een pandas dataframe wordt de nieuwe dataframe aangemaakt waarna deze over de oude dataframe opgeslagen wordt.

6.4. Sliding window

Het toevoegen en uitvoeren van analyses elke 15 frames is een verbetering tegenover de berekeningen elke frame uitvoeren, de Savitzky-Golay filter kan ook het gewenste resultaat leveren wanneer deze toegepast wordt op de X laatste datapunten. Hierdoor moeten er elke keer een kleinere, vaste hoeveelheid bewerkingen uitgevoerd worden waardoor de CPU minder belast zal worden.

6.5. TensorFlow Lite

Het ML model zelf kan ook omgezet worden naar een TFLite model, deze modellen hebben gelijkaardige resultaten maar nemen minder ruimte in en werken sneller. TFLite modellen zijn aangeraden wanneer er gewerkt wordt met mobiele apparaten of toestellen met beperkte rekenkracht.

6.6. Scope van het onderzoek

Omdat het implementeren van deze code in een mobiele applicatie niet binnen de scope van dit onderzoek ligt, is deze code niet uitgevoerd met live videobeelden. Het registreren van de repetities en het berekenen van de features kan echter wel in real-time, waardoor dit script kan dienen als proof of concept. Een vervolgonderzoek zou zich kunnen richten op de integratie van de vooraf vermelde technieken in een mobiele applicatie.

7

Resultaten

De getrainde modellen vertonen verschillende resultaten in het voorspellen van de RIR, het is belangrijk om niet alleen te kijken welk model het in het algemeen goed doet aangezien dit onderzoek zich vooral richt op het accuraat voorspellen van het punt waarin de sporter aan 0-2 RIR is.

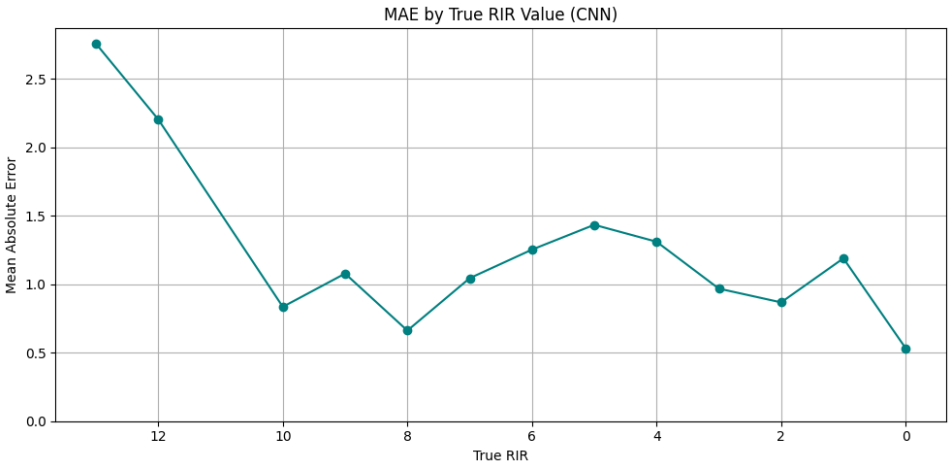
7.1. MAE en RMSE

De resultaten van het evalueren van de modellen worden uitgedrukt met de MAE (Mean Absolute Error) en de RMSE (Root Mean Squared Error).

Model	MAE	RMSE
RandomForestRegressor	1.040	1.258
GradientBoostingRegressor	0.982	1.194
XGBRegressor	0.936	1.174
CNN	1.072	1.319

Tabel 7.1: Resultaten evaluatie modellen.

Gebaseerd op deze tabel lijkt het dat het CNN geen nauwkeurige voorspellingen kan maken, door de custom 'loss' functie is het CNN echter nauwkeuriger dan de lineaire regressiemodellen bij de lage RIR. Een goede manier om het verschil tussen de modellen duidelijk te maken is met een visuele voorstelling:

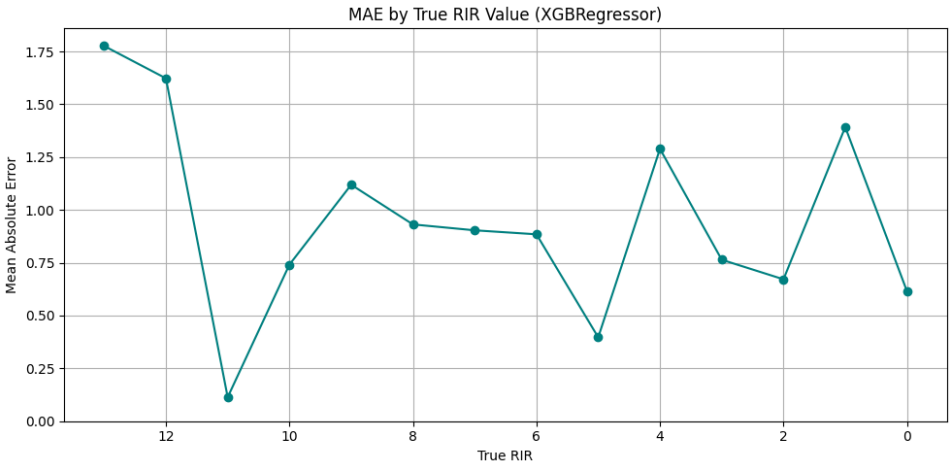


Figuur 7.1: MAE per RIR voor het CNN model.

Hier is duidelijk dat het model nauwkeuriger wordt naarmate de sporter dichter bij spierfalen komt, de hogere waardes zijn zeer onnauwkeurig waardoor het model onnauwkeuriger lijkt tegenover de lineaire regressiemodellen wanneer we enkel kijken naar de MAE. Het CNN model vertoont echter gelijkaardige resultaten als de andere modellen desondanks de kleine dataset. Het stijgen van de MAE bij 1 RIR heeft enkele mogelijke verklaringen:

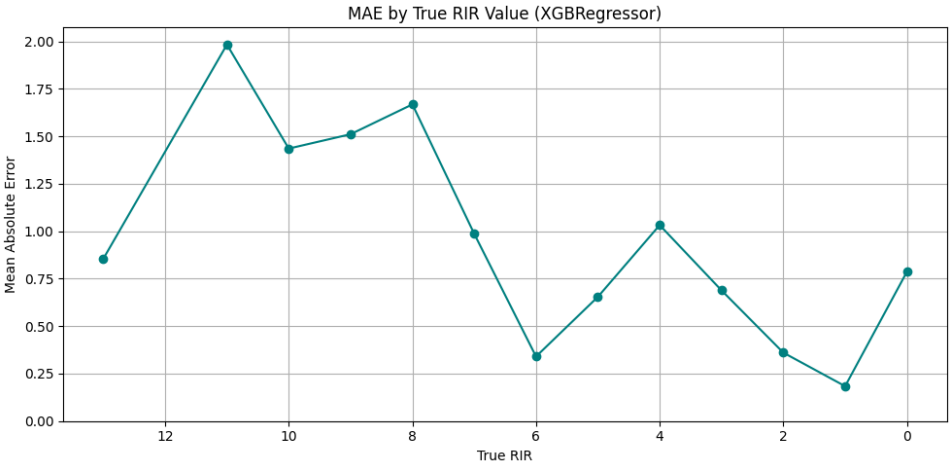
- Techniek van de sporter
- Detecteren van de repetitie
- Verband tussen snelheid en RIR

Het is mogelijk dat door vermoeidheid de uitvoering van de oefening veel begint te variëren van sporter tot sporter waardoor de modellen geen nauwkeurige voorspelling kunnen maken. Doordat er een groot verschil is in snelheid tussen 1 RIR en 0 RIR gaat de nauwkeurigheid weer omhoog na 1 RIR. Het is ook mogelijk dat doordat de laatste repetities veel vertragen in het laatste deel van de oefening dat het einde van de repetitie foutief aangeduid is. Dit is ook een gevolg van sporters die niet op dezelfde manier vermoeidheid vertonen. Het verband tussen de snelheid waarmee de repetitie uitgevoerd wordt en de RIR kan ook minder duidelijk zijn bij 1 RIR waardoor er extra aandacht aan deze repetitie moet gegeven worden bij het trainen van een model.



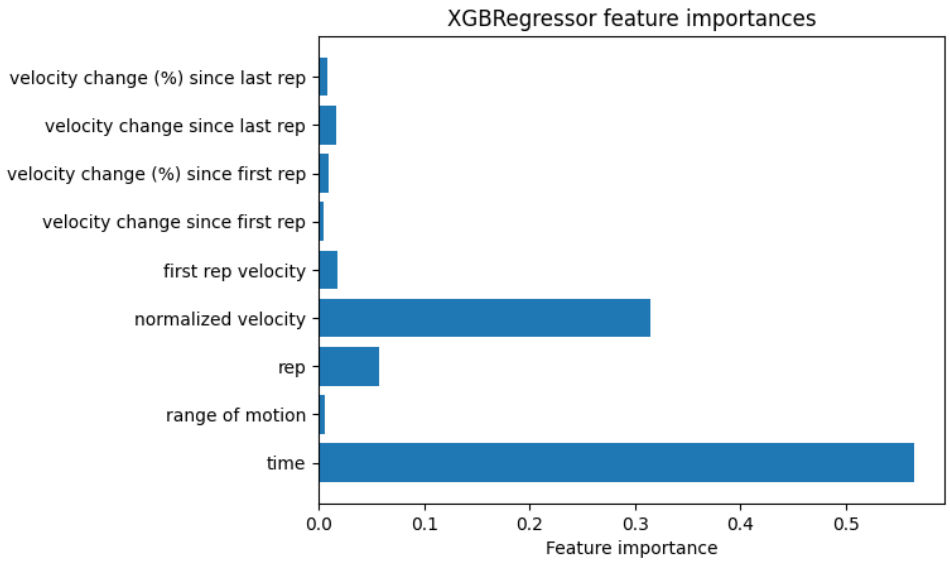
Figuur 7.2: MAE per RIR voor het XGBRegressor model.

Het regressiemodel heeft in het algemeen een lagere MAE dan het CNN model, maar dit komt vooral omdat het regressiemodel het beter doet bij de eerste repetities in een set (12+ tot 5 RIR). De onnauwkeurigheid rond 4 en 1 RIR die te zien valt in de evaluatie van het CNN model zijn ook hier te zien. Omdat deze onnauwkeurigheden mogelijks verklaard worden door variatie in techniek van sporters kan het nuttig zijn om een model te trainen op de data van 1 persoon. De MAE per RIR die horen bij dit model zijn de volgende:



Figuur 7.3: MAE per RIR voor het XGBRegressor model getraind op de data van 1 persoon.

De onnauwkeurigheid rond 1 RIR is volledig verdwenen, rond 4 RIR is het model nog steeds niet nauwkeurig. Een interessante verandering is het verschil in feature importance wanneer het model getraind is op de data van 1 persoon: de tijd waarin de repetitie uitgevoerd werd is nu de belangrijkste feature, wanneer de data van verschillende sporters gebruikt werd, was bij de lineaire regressiemodellen enkel de snelheid een relevante feature.



Figuur 7.4: Feature importance wanneer het model getraind is op de data van 1 persoon.

8

Conclusie

De modellen voorspellen de RIR van een sporter nauwkeuriger dan de sporter zelf, de manier waarop de data verwerkt is is een geschikte manier om verder modellen te trainen en kan op grotere schaal uitgewerkt worden in een volwaardige applicatie.

8.1. Nauwkeurigheid

Lineaire regressiemodellen voorspellen met minimale hyperparameter tuning op een nauwkeurige manier doorheen de hele set de RIR, bij deep learning modellen zoals CNNs moet er extra aandacht besteed worden aan de hyperparameters maar deze kunnen beter aangepast worden om bij de gewenste repetities de nauwkeurigheid te verhogen. Doordat het verband tussen de snelheid en de RIR bij het uitvoeren van de barbell bench press oefening met een gewicht met vaste verhouding tot de 1RM uitgesproken is, is beperkte data genoeg voor lineaire regressiemodellen.

8.2. Beïnvloedende factoren

In de onderzoeksvraag werd vermeld dat de invloed van kenmerken van de sporter onderzocht zou worden, in het onderzoek zelf viel op dat de techniek waarmee sporters de barbell bench press oefening uitvoeren genoeg variatie vertoont dat het trainen op data van 1 persoon zelfs binnen gelijkaardige categorieën een duidelijke verbetering in nauwkeurigheid als gevolg had. Deze verbetering was vooral te zien bij 1 RIR, waar sommige sporters de repetitie op een gelijkaardige manier aan 2 RIR of 0 RIR uitvoeren, dit kan dus moeilijk veralgemeend worden door het model.

8.3. Kwaliteit van de data

Het verschil tussen de eigenlijke en de door de sporter ingeschatte RIR heeft geen invloed op het verzamelen van de data. Sets kunnen tot spierfalen uitgevoerd worden waarna de data verwerkt kan worden op een manier waarop machine learning modellen getraind kunnen worden. Het verzamelen van de data zelf vraagt werk maar de bekomen data is van goede kwaliteit.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

Samenvatting

Deze bachelorproef heeft als doel om het aantal reps in reserve (RIR) van een sporter tijdens een set repetities in weerstandstraining te voorspellen. De doelgroep van dit onderzoek zijn ontwikkelaars die gebruik kunnen maken van dit model in applicaties, voornamelijk ontwikkelaars van fitness applicaties of onderzoekers die het aantal RIR als variabele willen gebruiken in verder onderzoek. Het verwachte resultaat is een model dat het aantal RIR nauwkeurig kan voorspellen in zowel ervaren als onervaren sporters. De methodologie die hiervoor toegepast zal worden begint met het kiezen van een toepasselijk model dat met verzamelde data getraind zal worden. De data waarmee gewerkt zal worden is het traject van de barbell bij het uitvoeren van barbell bench press sets. Na het trainen van het model zal de nauwkeurigheid hiervan geëvalueerd worden. Indien het model nauwkeurige voorspellingen kan maken kan het geïmplementeerd worden in fitness applicaties om verder een meer betaalbaar alternatief voor personal coaches aan te bieden of een beter gepersonaliseerd revalidatieplan op te stellen na blessures of ongevallen.

A.1. Inleiding

Weerstandstraining is een belangrijk aspect in het bouwen van spiermassa en kracht. De beschikbaarheid van commerciële sportscholen maakt individueel trainen op eigen gekozen uren een aantrekkelijke optie voor de doorsnee sporter om fysieke activiteit te hebben doorheen de week. De vrijheid die komt met het kiezen van eigen uren en oefeningen brengt echter ook problemen met zich mee. Het

feit dat begeleiding optioneel is zorgt ervoor dat velen zelf onderzoek doen naar optimale workouts, en voor het grootste deel is dit gelijkaardig aan de service die een personal coach zou leveren. Het grootste probleem waar beginners mee zitten wanneer ze een online schema volgen is de term 'reps in reserve', afgekort RIR: trainen tot spierfalen betekent dat er 0 RIR zijn, maar door de vermoeidheid die hieraan vasthangt is het niet aangeraden om dit herhaaldelijk te doen. Wanneer sporters zich hierdoor beginnen in te houden om de optimale 1-2 RIR te houden schatten ze dit vaak verkeerd in. Een mogelijke oplossing zou een applicatie zijn die aan de hand van videobeelden een voorspelling maakt van de hoeveelheid RIR na een set. De sporter krijgt zo feedback die het mogelijk maakt om het trainen verder te optimaliseren. Met welke nauwkeurigheid kan een applicatie in realtime de RIR van een sporter inschatten? Er zullen verschillende factoren de nauwkeurigheid van deze voorspellingen beïnvloeden. Zo kan een verschil in bijvoorbeeld leeftijd, geslacht of lengte invloed hebben op de manier waarop de oefeningen uitgevoerd worden. Op welke manier moet de data geclassificeerd worden om de nauwkeurigheid te maximaliseren? De techniek van sporters met weinig ervaring zal ook minder regelmatig zijn dan van ervaren sporters. Hoeveel impact heeft een verschil in ervaring tussen twee sporters? Als dit verschil te groot is zullen er extra stappen gezet moeten worden om een model te trainen. Een andere factor kan zijn dat de verzamelde data niet correct geclassificeerd is. Zoals eerder vermeld zijn sporters slecht in het inschatten van hun RIR. Wanneer we het model trainen moeten we zeker zijn dat de sporters hun RIR correct hebben ingeschat. Hoe kunnen we fouten met de data minimaliseren?

A.2. Literatuurstudie

A.2.1. 1RM

De bestaande literatuur rond dit onderwerp richt zich eerder tot het voorspellen van het meeste gewicht waarmee 1 repetitie uitgevoerd kan worden (de 'one rep max' of 1RM). Hierbij wordt de snelheid van de barbell bij het uitvoeren de oefening met lichtere gewichten onderzocht. De uiteindelijke conclusie is dat de snelheid van het uitvoeren van de oefening met lichtere gewichten een goede indicator is van de 1RM (Babault & Sekulic, 2021). De conclusie dat de snelheid van een repetitie gebruikt kan worden om met voldoende nauwkeurigheid het maximale gewicht te voorspellen is een goede indicator dat hiermee ook het maximale aantal repetities voorspeld zal kunnen worden.

A.2.2. Model

Bij een eerder onderzoek (Jukic e.a., 2024) werd duidelijk dat het verband tussen snelheid van de repetitie en RIR voorgesteld kan worden met regressie. Aangezien de data in dit eerder onderzoek echter verzameld is in een labo zal het voorspellen

van de RIR in real-time aan de hand van een applicatie waarschijnlijk extra complicaties met zich meebrengen.

A.2.3. Foutmarge getrainde proefpersonen

Een relevante studie over de nauwkeurigheid waarmee getrainde proefpersonen de RIR inschatten toont dat zelfs bij getrainde personen er een foutmarge van 1 repetitie zit (Refalo, 2023). Een model dat nauwkeuriger RIR kan voorspellen zou dus ook voor ervaren sporters een toegevoegde waarde kunnen leveren.

A.2.4. Tracen van barbell traject

De derde relevante literatuur is het bewijs dat mobiele applicaties in staat zijn het traject van een barbell tijdens een oefening correct vast te leggen (Balsalobre-Fernández e.a., 2020). Hieruit blijkt dat een applicatie om de RIR te voorspellen bruikbaar zou zijn in commerciële sportscholen en dat eigen beelden voldoende nauwkeurig zijn als data.

A.3. Methodologie

Om het onderzoek te starten moet er eerst een model gekozen worden dat getraind zal worden met data, dit komt waarschijnlijk neer op het kiezen tussen Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTMs) of Convolutional Neural Networks (CNNs). Hierna zal er data verzameld worden in de vorm van videobeelden van proefpersonen met variërende ervaring met weerstandstraining waarbij een set tot spierfalen uitgevoerd wordt. Het feit dat de data eindigt met spierfalen betekent dat we gemakkelijk data halen van een set met nog 1 of 2 repetities in reserve, afhankelijk van waar de beelden gestopt worden. De data zal bestaan uit 3 sets van verschillende vrijwilligers in een commerciële sportschool, deze data zal verzameld worden over de tijdspanne van een week. Bij enkele proefpersonen zal extra data verzameld worden zodat de voor- en nadelen van een uniek model per persoon onderzocht kunnen worden. De eigenlijke data waarmee het model zal werken is het resultaat van het verwerken van de videobeelden waarbij het traject van de barbell onderzocht wordt aan de hand van computer vision. Omdat de barbell bench press een oefening is waarbij de barbell stabiel blijft en er weinig variatië zit op de richting die de barbell aflegt zullen we deze oefening kiezen om het onderzoek op uit te voeren. Bij het verwerken van de data zal onderzocht worden welke factoren het meeste invloed hebben op de nauwkeurigheid van het model. Indien blijkt dat sommige factoren geen of verwaarloosbare invloed hebben kan het model veralgemeend worden, blijkt echter dat sommige factoren veel invloed hebben zullen er om de nauwkeurigheid te maximaliseren verschillende modellen opgesteld moeten worden. De voornaamste factoren die onderzocht zullen worden zijn geslacht en leeftijd van de sporter, ervaring van de sporter en aantal voorgaande oefeningen in de workout. Wanneer de data verwerkt is kan

deze gebruikt worden om het gekozen model te trainen. Hierna kunnen er nieuwe, ongeziene, beelden gebruikt worden om het model te testen en kan de nauwkeurigheid van het model geëvalueerd worden.

A.4. Verwacht resultaat, conclusie

Het verwachte resultaat is dat het bekomen model nauwkeurig genoeg voorspellingen maakt om van toepassing te zijn in applicaties. Een mogelijk struikelblok is dat het verschil tussen onervaren en ervaren sporters het trainen van het model moeilijk kan maken. Indien de nauwkeurigheid van het model hoog genoeg is kan het verband tussen de snelheid van een repetitie en het aantal RIR ook visueel voorgesteld worden.

Bibliografie

Babault, N. & Sekulic, D. (2021). *Use of Machine-Learning and Load–Velocity Profiling to Estimate 1-Repetition Maximums for Two Variations of the Bench-Press Exercise*. https://pmc.ncbi.nlm.nih.gov/articles/PMC8002214/

Balsalobre-Fernández, C., Geiser, G., Krzyszkowski, J. & Kipp, K. (2020). *Validity and reliability of a computer-vision-based smartphone app for measuring barbell trajectory during the snatch*. https://pubmed.ncbi.nlm.nih.gov/32079484/

Jukic, I., Prnjak, K., Helms, E. R. & McGuigan, M. R. (2024). *Modeling the repetitions-in-reserve-velocity relationship: a valid method for resistance training monitoring and prescription, and fatigue management*. https://pmc.ncbi.nlm.nih.gov/articles/PMC10901726/#phy215955-sec-0002

Refalo, M. (2023). *Accuracy of Intraset Repetitions-in-Reserve Predictions During the Bench Press Exercise in Resistance-Trained Male and Female Subjects*. https://www.researchgate.net/publication/375687905_Accuracy_of_Intraset_Repetitions-in-Reserve_Predictions_During_the_Bench_Press_Exercise_in_Resistance-Trained_Male_and_Female_Subjects