Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Computer Science	Vårsemesteret, 2017 Åpen / ~~Konfidensiell~~
Forfatter: Andreas Hove	…… (signatur forfatter)
Fagansvarlig: Hein Meling Veileder(e): Hein Meling, Ståle Freyer, Arild Eikefjord (Laerdal Medical)	
Tittel på masteroppgaven: Kvalifisering av smartklokke til bruk i HLR Engelsk tittel: Qualifying smartwatches for use in CPR	
Studiepoeng: 30	
Emneord: Hjerte-Lunge-Redning, Lineær-filtrering, Maskinlæring, Android, TensorFlow	Sidetall: 48 Stavanger, 15/06/17 dato/år

Kvalifisering av smartklokke til bruk i HLR

Andreas Hove

Institutt for data- og elektroteknikk
Det teknisk-naturvitenskapelige fakultet
Universitetet i Stavanger
2017

Forord

Jeg vil rette en stor takk til professor Hein Meling ved Universitetet i Stavanger for hans idéer, motivasjon og verdifulle tilbakemeldinger til oppgaven. Jeg vil også takke sjefingeniør Ståle Freyer ved Universitet i Stavanger for teknisk assistanse og tilgjengelighet gjennom semesteret. Til slutt vil jeg takke min samboer, Marianne, for tålmodighet og god støtte, og min nyfødte datter Mathea, for hennes oppmuntrende skrik.

Sammendrag

I denne oppgaven ble en smartklokke benyttet til å analysere brystkompresjoner ved hjerte-lunge-redning (HLR). Det ble utviklet to metoder for å måle dybde og rate i brystkompresjoner. Metodene ble verifisert i to eksperiment; ved manuelle kompresjoner på en gjenopplivningsdukke og ved automatiske kompresjoner utført av en industrirobot. Det ble utviklet en smartklokke-applikasjon som implementerte metodene og kan veilede brukeren til utførelse av god HLR. Resultater fra eksperimentene viser at en smartklokke kan benyttes til veiledning, men nøyaktigheten til metodene kunne ikke verifiseres. Det antas at metodene vil gi tilfredsstillende resultater ved et utvidet eksperiment.

INNHOLD

1 INNLEDNING

1.1 Motivasjon

Ved hjerte-lunge-redning (HLR) er målet å komprimere brystkassen for å få hjertet til å pumpe blod til hjernen. Perfekte kompresjoner gir en blodsirkulasjon på rundt 30 prosent av det volumet et friskt hjerte sirkulerer. Ved dårlige kompresjoner reduseres denne andelen betydelig, og pasientens sjanse for å overleve faller drastisk [1]. For å utvikle lekfolks kunnskap og trygghet i utførelse av HLR, må de ha tilgang til god opplæring og jevnlig trening. Dette er vanskelig for folk å prioritere da slik trening kan være dyrt og tidkrevende. Dette kan føre til at mange utsetter den nødvendige opplæringen. Når uhellet først er ute, vil en kvalifisert smartklokke-applikasjon kunne styrke lekfolks kvalitet på HLR og potensielt redde liv. Ved å utvikle en applikasjon (app) til smartklokker utvikles en digital HLR assistent; et godt eksempel på å utnytte verdien av biomedisinsk informatikk. En person som arbeider med assistanse fra en informasjonsressurs er bedre enn den samme personen uassistert [2].

1.2 Oppgaven

I dette avsnittet beskrives oppgavens målsetninger. Oppgaven gikk ut på lage en smartklokke-app som assisterer ved utførelse av HLR. Det ble utviklet en app som evaluerer kvaliteten på HLR-kompresjoner i sanntid, og gir brukeren visuelle og fysiske tilbakemeldinger. Det ble implementert to algoritmer i appen for å evaluere kompresjonsdybde, en algoritme for lineær filtrering (LF) og en maskinlæringsalgoritme (ML). LF-algoritmen beregner kompresjonsdybde fra akselerometeret i smartklokken, mens ML-algoritmen benytter både akselerometeret og gyroskopet. Det ble i tillegg implementert en algoritme for å beregne kompresjonsraten. For å verifisere algoritmenes nøyaktighet ble de testet ved bruk av en industrirobot fra ABB til å utføre kompresjoner, og ved manuelle kompresjoner på en gjenopplivningsdukke. Det ble gjort en vurdering av om sensorene i smartklokken og algoritmene gir tilstrekkelig nøyaktighet på tilbakemeldingene, for å benyttes til veiledning av en livredder.

Utstyr	Supplert av
Smartklokke	Laerdal Medical
Gjenopplivningsdukke	Laerdal Medical
ABB industrirobot	Universitetet i Stavanger
3D-printet håndledd	Universitetet i Stavanger

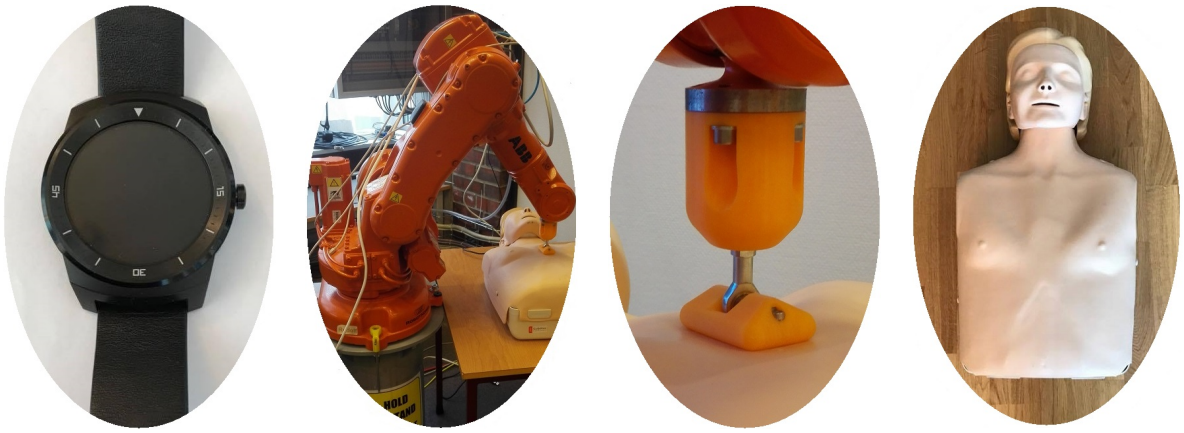
Tabell 1: Utstyr

1.3 Utstyr

I dette avsnittet beskrives utstyret som ble valgt. I dagens smartklokker finnes det ofte flere sensorer for å kunne nøyaktig måle akselerasjon, rotasjon, trykk og temperatur. Ettersom klokken erfarer akselerasjon og rotasjon i tre dimensjoner, trengs et tre-akse akselerometer og et tre-akse gyroskop. Følgende kriterier ble tatt hensyn til ved valg av smartklokke:

- Tre-akse akselerometer
- Tre-akse gyroskop
- Høy samplingsrate på sensorene
- Et operativsystem som har god dokumentasjon samt ressurser på Internett.

LG G Watch R ble valgt da denne oppfyller kriteriene over. Klokken kan lese akselerometeret med en rate på 200 Hz. Klokken kjører operativsystemet Android Wear OS som er en versjon av det populære operativsystemet Android. Denne versjonen er rettet mot smartklokker og andre håndholdte enheter. Det skal derfor utvikles en Android-app i denne oppgaven. Klokken og resten av utstyret er vist i figur 1.



(a) Smartklokke (b) ABB Industrirobot (c) 3D-printet håndledd (d) Gjenopplivningsdukke

Figur 1: Utstyret som skal benyttes

Gjenopplivningsdukken (Little Anne, Laerdal Medical) etterligner en menneskelig overkropp. Dukken inneholder en motstandsfjær som har en fjærkraft på 500 Newton fullt komprimert. Dukken inneholder også en optisk sensor for dybdemålinger, som enten kan kobles til en PC via USB eller en smarttelefon via Bluetooth.

Det ble benyttet en industrirobot fra ABB til å teste appen. Sensordata ble samtidig hentet ut fra klokken. Roboten ble programmert ved hjelp av programmeringsspråket Rapid som er et høy-nivå språk utviklet av ABB for robotprogrammering. Rapid-koden ble skrevet i det integrerte utviklingsmiljøet RobotStudio. Fordelene med å benytte en robot er:

- Presisjon i målingene
- Mulighet for å gjenta målinger med høy presisjon

- Tidsbesparende sammenlignet med å utføre store mengder manuelle kompresjoner

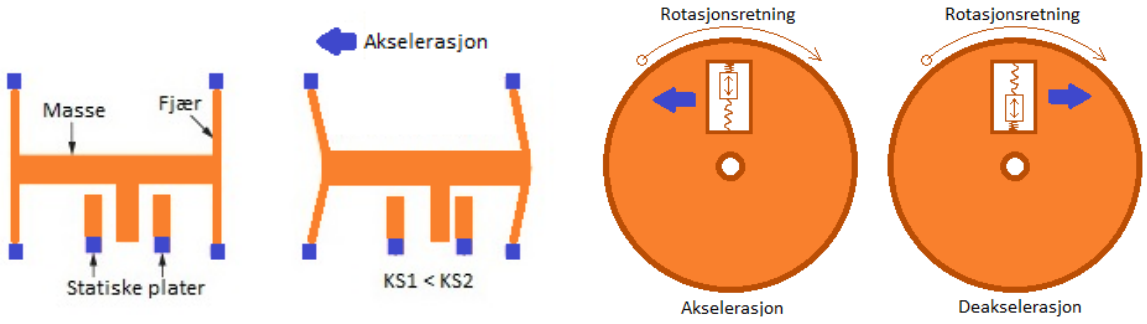
Laptopen som ble brukt er en Lenovo T460 med 8GB RAM og en Intel i5 prosessor (2.4 GHz). Av programvare ble utviklingsmiljøet Android Studio benyttet for utvikling av Android-appen som forøvrig kodes i programmeringsspråket Java. For utvikling av ML-algoritmen samt dataprosessering ble Python og tilhørende utviklingsmiljø PyCharm benyttet. MATLAB ble brukt til å utvikle algoritmene og filtrene i kapittel 2.2. Github ble brukt til versjonskontroll og sikkerhetskopiering av kildekode.

1.4 Sensorer

I dette avsnittet beskrives hvilke sensorer i smartklokken som ble benyttet. Det finnes ulike typer akselerometer og gyroskop, der klokkens sensorer er mikroelektromekaniske systemer (MEMS).

Akselerometeret fungerer som illustrert i figur 2a. Når den påføres en ekstern kraft flyttes den indre massen og det oppstår en endring i kapasitansen, $\Delta KS = KS2 - KS1$. En krets, som oftest bygget på samme chipen, omgjør kapasitansen til spenning [3]. Denne spenningen blir konvertert til digitale signaler ved hjelp av en A/D-konverter.

Når gyroskopet roteres vil den indre massen flyttes fordi vinkelhastigheten endres, illustrert i figur 2b. Tilsvarende til akselerometeret skaper dette en endring i kapasitansen som omgjøres til målbar spenning.



(a) Akselerometer

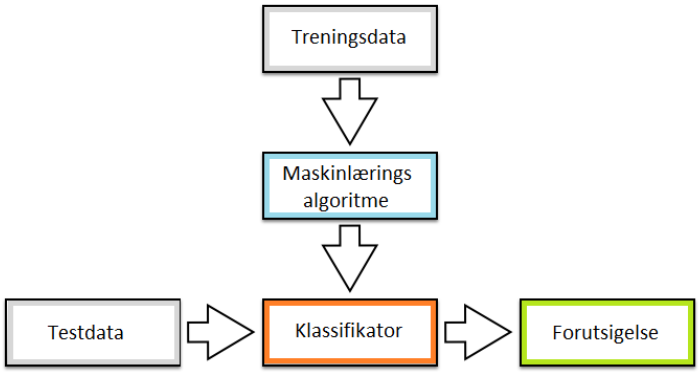
(b) Gyroskop

Figur 2: Illustrasjon av sensorene på mekanisk nivå

1.5 Maskinlæring

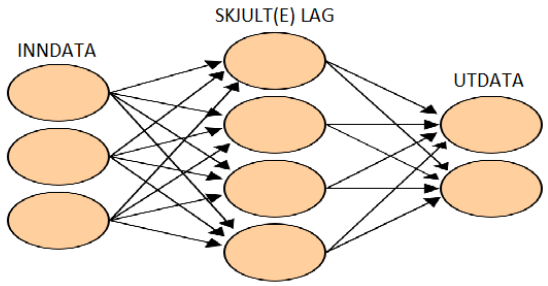
I dette avsnittet forklares teorien bak maskinlæring og valget av ML-algoritme. Maskinlæring er en underkategori av kunstig intelligens. Maskinlæring deles grovt opp i veiledet læring og ikke-veiledet læring. Ikke-veiledede modeller lærer seg data ved å gruppere de som har liknende egenskaper. Veiledede modeller trenger ikke å gruppere da inndata allerede er merket med en kategori som de tilhører. Denne oppgaven benytter veiledet læring. En algoritme

for veiledet læring lager en klassifikator som kan motta nye usette data, og deretter avgjøre hvilken kategori de hører til, vist i figur 3. En mer generell betegnelse på klassifikator er modell, som benyttes videre i oppgaven.



Figur 3: Oversiktsdiagram maskinlæring

Maskinlæring har flere underkategorier og kunstige nevrale nettverk (NN) er en av dem. Det finnes mange ulike NN og noen av de vanligste er som følger. Mate-fremover NN er et nettverk der data flyter kun én vei, fra inngangsnevron mot utgangsnevron, vist i figur 4. Motsatt er tilbakevendende NN, et nettverk der data kan flyte begge veier. Modulære NN består av flere uavhengige NN som jobber side om side, der utdata fra alle blir kombinert til et resultat. Fysiske NN er et fysisk nettverk avhengig av elektronisk justerbar resistans for å simulere et NN.



Figur 4: Strukturen til et mate-fremover nevralt nettverk

1.5.1 Oppbyggingen av et nevralt nettverk

Et NN er et system som etterligner biologiske nervevev. Det finnes ulike algoritmer for å implementere NN, i denne oppgaven ble det implementert et konvolusjonelt nevralt nettverk (KNN) som er et mate-fremover NN. Arkitekturen til KNN er inspirert av dyrenes synssenter bak i hjernen. Der blir nye sanseinntrykk sammenlignet med tidligere erfarte inntrykk, og

det er denne tolkningen som gir det nye inntrykket mening. Det ble valgt å implementere et KNN da slike nett har vist god nøyaktighet på å gjenkjenne menneskelige aktiviteter fra sensordata [4]. Egenskapen til slike nettverk er at de er selvlærende etter at de blir eksponert for treningsdata med tilhørende kategori. Når NN-algoritmen er trent opp, blir det produsert en modell. Den trente modellen blir eksponert for ukjente testdata, med samme struktur som treningsdata, for så å forutsi hvilken kategori de tilhører. Et NN er bygget opp av følgende komponenter:

- Aktiveringsfunksjonen er det som gjør et NN ikke-lineært og gir nettet muligheten til å lære underveis, ellers ville det kun utført samme kalkulasjon for samme inndata hver gang.
- Et NN består av flere nevroner. Hvert nevron kommuniserer med en mindre del av de andre nevronene i nettverket. Alle nevronene får ett sett med inndata, utfører en kalkulasjon og produserer utdata. Informasjonen beveger seg gjennom nettet helt til den når endenevronet, som har produsert informasjon om hvilken kategori inndataene tilhører.
- I koblingen mellom to nevroner er det kalkulert en vekt. Denne bestemmer hvor mye utdata fra nevron A har å si på inndata til nevron B. Det er disse vektene som oppdateres når modellen trenes.
- En bias er en konstantverdi for hver kobling mellom to nevroner. Denne er ofte tilkoblet vekten.
- Å velge antall skjulte lag er en del av å utvikle en god modell. Det er ingen tommelfingerregel, men generelt er dypere nettverk alltid bedre, men krever mer data og økt kompleksitet i læringen.

1.6 Matematiske konsepter

Det finnes flere metoder for å beregne dybde ut fra akselerasjonsdata. I LF-metoden ble beregningene utført ved integrasjon i tidsdomenet. Akselerasjon er den første deriverte av hastighet, og hastighet er den første deriverte av strekning [5]. Dermed kan akselerasjon dobbelintegreres for å få strekning med likningene vist i likning (1) og (2), der a(t) er akselerasjon, v(t) er hastighet og s(t) er strekning, alle variablene som funksjon av tid.

$$v(t) = v(0) + \int_0^t a(t)\,dt \tag{1}$$

$$s(t) = s(0) + \int_0^t v(t)\,dt \tag{2}$$

I akselerasjonsmålinger er det mange faktorer som spiller inn og skaper støy; instrumentell ustabilitet, bakgrunnsstøy og grunnverdiene som varierer fra sensor til sensor. Disse støymålingene dobles ved integrering, og firedobles ved dobbelintegrering.

Sensoren har tre registre for akselerasjonsdata, ett for hver av aksene, x, y og z. Akselerasjonen for hver akse presenteres ved følgende variabler, a_x, a_y og a_z. For å finne den totale akselerasjonen i en kompresjon ble dataene normert til en skalar vist i likning 3.

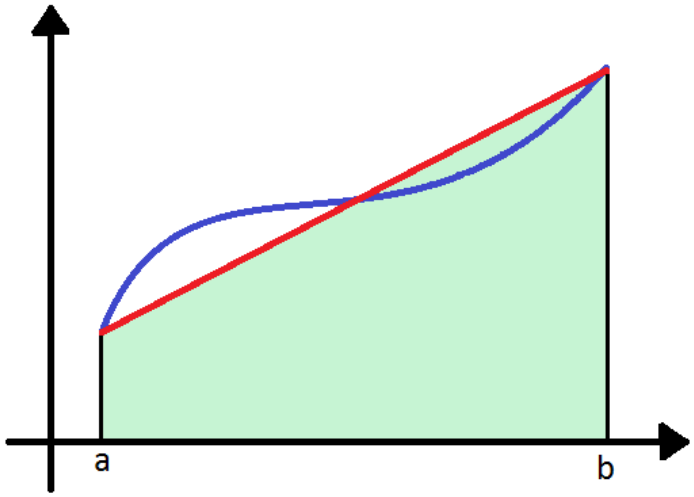
$$a_{norm} = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3}$$

I motsetning til å beregne dybden på hver enkelt akse, der gravitasjon vil summere opp til 1 g, kan 1 g simpelthen subtraheres fra a_{norm} for å få en akselerasjon tilnærmet null når sensoren er i ro eller ved konstant hastighet.

1.6.1 Integreringsmetode

Integrering av akselerasjon og hastighet ble utført ved trapesmetoden. Den estimerer integralet ved å kalkulere arealet under en graf, ved likning 4. Dette arealet er vist i lysegrønt i figur 5. Denne metoden er ekstremt nøyaktig for periodiske signaler [6], og er derfor velegnet til analyse av brystkompresjoner.

$$\int_a^b f(t)\,dt \approx (b-a)\left[\frac{f(a)+f(b)}{2}\right] \tag{4}$$



Figur 5: Trapesmetoden for estimering av integral

1.7 Bakgrunn

1.7.1 Hjerte-lunge-redning

Igangsetting av HLR er den viktigste delen av behandlingen ved akutt hjertestans, som årlig krever livet til cirka 2500 nordmenn. Mange dør fordi HLR ikke blir igangsatt i tide [7]. Dagens løsning for assistanse ved akutt hjertestans er å motta verbal veiledning fra AMK på telefon. Dette gir lite kvalitetssikker HLR, helsepersonell hos AMK kun har innsyn i situasjonen ut fra det innringer formidler.

HLR er kritisk for overlevelse og for minimering av hjerneskade ved akutt hjertestans. I Norge mottar 75 prosent av hjertestans-ofre HLR før nødpersonell ankommer. Av disse 75 prosentene får $\frac{1}{3}$ igjen sin normale hjerterytme før ambulansen kommer til stedet, ved at pasienten gjenopplives ved bruk av en hjertestarter [8].

En hjertestarter er i de fleste tilfeller nødvendig for gjenopplivning. Den gir elektrisk sjokk via elektroder som festes på kroppen som skal få hjertet tilbake i normal rytme. I ekstremt få tilfeller får pasienten tilbake hjerterytmen kun ved HLR. Lekfolk tar som oftest i bruk en automatisk hjertestarter som vil gi sjokk i tilfeller der pasienten har rytmeforstyrrelser som kan sjokkbehandles.

Mange lekfolk tør ikke utføre HLR fordi de er usikre på hvordan det gjøres og er redd for å gjøre skade. De har en viss kjennskap til utførelsen av HLR, men ikke nødvendigvis korrekt teknikk. Med riktig trening kan de redde liv, men det er de færreste som har den nødvendige mengdetreningen for å utføre kvalitets-HLR.

1.7.2 Gjeldende retningslinjer

I dette avsnittet beskrives de gjeldende retningslinjene for HLR fra 2015 [9]. Før HLR igangsettes må en forsikre seg om at pasienten er livløs. Først skal en sjekke bevissthet ved å riste i og snakke tydelig til pasienten. Dersom det ikke er respons må en gjøre følgende:

1. Sørg for frie luftveier
2. Sjekk om pasienten puster normalt
3. Dersom pasienten ikke puster og er bevisstløs: ring 113 og start HLR.

Den anbefalte posituren for brystkompresjoner er å knele ved pasienten, med utstrakte armer plasseres den ene håndflaten midt på brystkassen og fingrene låses med den andre hånden. Videre er korrekt utførelse av HLR viktig for kvaliteten, der kompresjonsdybde og kompresjonsrate er de to viktigste faktorene. Hvor dypt brystet skal komprimeres varierer fra person til person, men generelt gjelder følgende:

- Baby: 38 mm
- Barn: 50 mm
- Voksne: 50 - 60 mm

Kompresjonsraten skal ligge mellom 100 og 120 kompresjoner per minutt (kpm). Det anbefales å komprimere i takt med den klassiske diskosangen "Stayin' Alive" av Bee Gees som tilsvarer cirka 100 kpm. Ved innblåsning skal en løfte opp haken og klemme igjen neseborene, blåse inn og se i sidesynet at brystkassen hever seg.

1.7.3 Manglende kunnskaper hos lekfolk

På grunn av usikkerhet i egne livredningsferdigheter velger mange å ikke utføre HLR i akutte situasjoner i påvente av et mer erfarent individ. Den gylne regelen er at all HLR er bedre enn ingen HLR. Blant annet har British Health Foundation begynt å kun anbefale kompresjoner og droppe innblåsingene, for å forenkle HLR og engasjere flere til å lære.

Frykten for å skade en person kan bli større enn viljen til å utføre HLR. Å briste eller brekke ribbein på en klinisk død pasient er relativt ufarlig, da alternativet til pasienten kan være døden. Skade på ribbein forekommer ofte ved HLR, i 29% av tilfellene [10]. Spesielt voksne og eldre får skader, mens bein hos barn er mer fleksible og brekker derfor ikke like lett. Livredderen kan begynne å komprimere svakere fordi et ribbein har brukket. Dette kan redusere kvaliteten fordi kompresjonene blir for grunne.

I tillegg til at mange er engstelig for å skade pasienten, er mange redd for å bli saksøkt. Heldigvis finnes lover som beskytter alle som gir grunnleggende livredning, i USA kalt Good Samaritan Law. I en undersøkelse utført av San Francisco Department of Public Health i 1996 ble det funnet at hovedårsaken til at forbipasserende ikke utfører HLR er frykt for at personen later som om de har hjerteinfrakt, og i realiteten skal rane dem. Det er også lover som beskytter pasienten; dersom den forbipasserende handler uaktsomt ved å forlate pasienten uten tilsyn, er det i de fleste land straffbart, deriblant her i Norge.

Retningslinjene for livredning har endret seg opp gjennom årene og er i dag en syklus på 30 kompresjoner etterfulgt av 2 innblåsninger (forkortet 30:2) [9]. Tidligere var både 15:1 og 15:2 anbefalt [11]. Den gjeldende retningslinjen har færre innblåsninger per minutt som reduserer opphold i kompresjoner sammenlignet med 15:2 og 15:1. Det er foreslått å droppe innblåsningene for å hindre oppholdet i kompresjoner [12], en metode som hadde positivt utfall i et forsøk på svin. I artikkelen anbefales det å kun utføre kompresjoner fordi det er en enkel prosedyre som de fleste husker og artikkelen mener at det fører til at flere utfører HLR. Artikkelen anbefaler derimot at personer som er trent opp i HLR skal følge 30:2 regelen. At retningslinjene oppdateres såpass ofte, cirka hvert femte år, skaper forvirring fordi folk ikke er flinke nok til å holde seg oppdatert.

1.7.4 Mulighetene med en HLR-assistent

Utøvelsen av HLR blant lekfolk har økt jevnt de siste tiårene og en dansk studie viser en økning fra 21.1 prosent i 2001 til 44.9 prosent i 2010 [13]. Selv om flere utfører HLR, betyr det ikke nødvendigvis at kvaliteten er god. De som blir kurset kan ofte ha flere års opphold mellom hvert kurs, som reduserer den oppnåelige kvaliteten. Ved at lekfolk mottar utvidet assistanse

ved livredningen, vil trolig flere utføre HLR fordi de får økt selvtillit og potensielt kan redde liv. Godt trent helsepersonell klarer heller ikke alltid å utføre kvalitets-HLR [14] og vil ha god nytte av slik assistanse.

Smartklokker har inntatt markedet de siste årene. Det ble solgt henholdsvis 5, 19 og 39 millioner smartklokker i 2014, 2015 og 2016 [15]. Salgstallene er forventet å dobles i 2017. Ved å utvikle en smartklokke-app vil lekfolk ha tilgang til et billig verktøy, uten behov for spesialutstyr. Usikkerhet og mangel på trening er noe en app til smartklokke kan gjøre noe med. Selv de minste oppgaver som å holde telling på antall kompresjoner er vanskelig i en hektisk og kritisk situasjon. Det vil fortsatt kreves av lekfolk at de får nødvendig trening i grunnleggende livredning; en app vil aldri erstatte kunnskap og trening fra et livredningskurs.

1.7.5 Eksisterende teknikker for å måle kompresjonsdybde

En mye brukt metode er dobbelintegrering av akselerasjon [14, 16, 17, 18, 19, 20]. En annen metode utfører spektralanalyse av akselerasjonsdata [14]. I spesialbygde enheter implementeres ofte trykksensor i tillegg til et akselerometer [21]. Trykksensoren benyttes for å avgjøre når kompresjonen starter og slutter, for å få et mer nøyaktig dybdemål grunnet ustabiliteten ved dobbelintegrasjon. Det finnes også en metode som ved å bruke mobilkameraet kan beregne kompresjonsrate i sanntid [22].

I et arbeid fra 2009 [23] ble det utviklet en metode for å beregne kompresjonsdybde i tre dimensjoner ved bruk av ett akselerometer og to gyroskop. Kompresjonsdybden ble kalkulert ved dobbelintegrasjon av akselerasjon, i tillegg til å kalkulere vinkelendringen i løpet av kompresjonen ved å bruke gyroskopet. Det kan så kompenseres for feilmålinger i kompresjonsdybden på grunn av vinkelendringene. Det ble ikke utført et omfattende eksperiment for metoden, men resultatene viser et utsnitt fra manuelle kompresjoner der dybden er beregnet innenfor anbefalte retningslinjer.

1.7.6 Eksisterende HLR-assistenter

I tillegg til apper, finnes HLR-assistenter som egne enheter. Blant annet CPRMeter fra Laerdal Medical måler kompresjonsdybde og kompresjonsrate i sanntid og gir tilbakemeldinger underveis og etter en fullført sesjon. Enheten plasseres på brystet til pasienten og ved hjelp av akselerometer og trykksensor beregnes dybden og raten. CPRMeter brukes i dag i reelle gjennopplivnings-situasjoner, ofte tilknyttet en hjertestarter. CPRMeter ble utviklet etter gode resultater i dette arbeidet fra 2002 [21].

Det finnes i 2017 en rekke apper som kan assistere ved akutt hjertestans. De forskjellige appene benyttes til å lokalisere livreddere og hjertestartere, instruere livredning og gi tilbakemeldinger på HLR-kvalitet. Mye brukt i oppgaven er Laerdal Medical sin QCPR app som gir sanntidstilbakemelding på kompresjonskvaliteten ved trening med dukke. Appen kobles til en av Laerdals gjenopplivningsdukker via Bluetooth og gir tilbakemelding på kompresjonsdybde og kompresjonsrate, der dybden kalkuleres fra dukkens innebygde optiske sensor.

Smarttelefon-apper som gir tilbakemeldinger i sanntid, har til felles at telefonen må holdes i hånden mens det komprimeres, og det kan være utfordrende å opprettholde grepet. Det er også tidkrevende da telefonen må legges ned når det skal gis innblåsninger. Alle appene har samme funksjonalitet, de gir

- brukeren informasjon om utførelsen av HLR
- hvordan og når en skal tilkalle hjelp
- sanntidstilbakemelding på kompresjonsdybde og kompresjonsrate

Det er tre apper som gir tilbakemeldinger i sanntid til smarttelefoner:

- ZOLL PocketCPR utviklet av ZOLL Medical
- BHF (British Health Foundation) PocketCPR utviklet av ZOLL Medical
- UCPR utviklet av MELab

Begge PocketCPR bruker samme algoritme men med noe forskjellig utseende. I tillegg kan PocketCPR appene ringe nødnummeret mens brukeren komprimerer. Appene for smartklokker har til felles at skjermen viser kompresjonsdybde og kompresjonsrate i tekst og med fargekoding på bakgrunnen. Dette tar mye plass på små skjermer og kan være vanskelig, om ikke umulig, å følge med på mens HLR utføres.

Det finnes foreløpig to smartklokke-apper med samme formålet som denne oppgaven. W-CPR utviklet av MELab har grunnlag i artikkelen til Yeongtak Song et al. [20]. W-CPR gir tilbakemeldinger tilsvarende smarttelefon-appene; kompresjonsdybde og kompresjonsrate i tekst og farger. Appen CPWeaR utviklet av Team SACK logger en kompresjonsesjon på smartklokken og sender data til smarttelefon som viser kompresjonsdybde og kompresjonsrate samt annen statistikk etter sesjonen er fullført. Appen gir ikke tilbakemeldinger i sanntid.

Felles for alle appene nevnt ovenfor er at de kun benytter akselerometer i kalkulasjonene. Det finnes i tillegg veldig mange apper som ikke gir tilbakemelding i sanntid, men kun gir generelle instrukser på hvordan HLR skal utføres. Det ble i arbeidet Tomohiko Sakai et al. [24] konkludert at apper som kun gir HLR-instrukser, ikke forbedrer brukerens komprimering.

HLR-assistenter har bevist god effekt på å forbedre lekfolks utførelse av HLR [20, 25], men koster ofte flere tusen kroner når det er snakk om et spesialisert verktøy. Ved å utvikle en app med kvalitetssikrede algoritmer, vil lekfolk ha tilgang til et lavkostnads assistanseverktøy. Utfordringen med smarttelefoner og andre HLR-enheter er at de er vanskelig å holde mens det utføres hjertekompresjoner. En smartklokke festet på håndleddet løser dette problemet.

Ved å kun bruke ett akselerometer er det kun ett referansepunkt. Dette kan ikke benyttes til å avgjøre om det er pasientens brystkasse som komprimeres, eller omgivelsene rundt pasienten som beveger seg. Akselerometerbaserte algoritmer er kun nøyaktige hvis brystkassen blir fullstendig dekomprimert etter hver kompresjon [18]. For å unngå dette kan ufullstendige kompresjoner registreres og brukeren varsles om å ikke lene seg på pasienten [26].

2 IMPLEMENTASJON

I dette kapitlet beskrives to metoder appen benytter for å beregne kompresjonsdybde og kompresjonsrate i sanntid. Den første metoden dobbelintegrerer smartklokkens akselerasjon. Den andre metoden implementerer et nevralt nettverk som trenes opp til å klassifisere kompresjoner ut fra sensordata. I tillegg beskrives den utviklede appen og det eksperimentelle oppsettet.

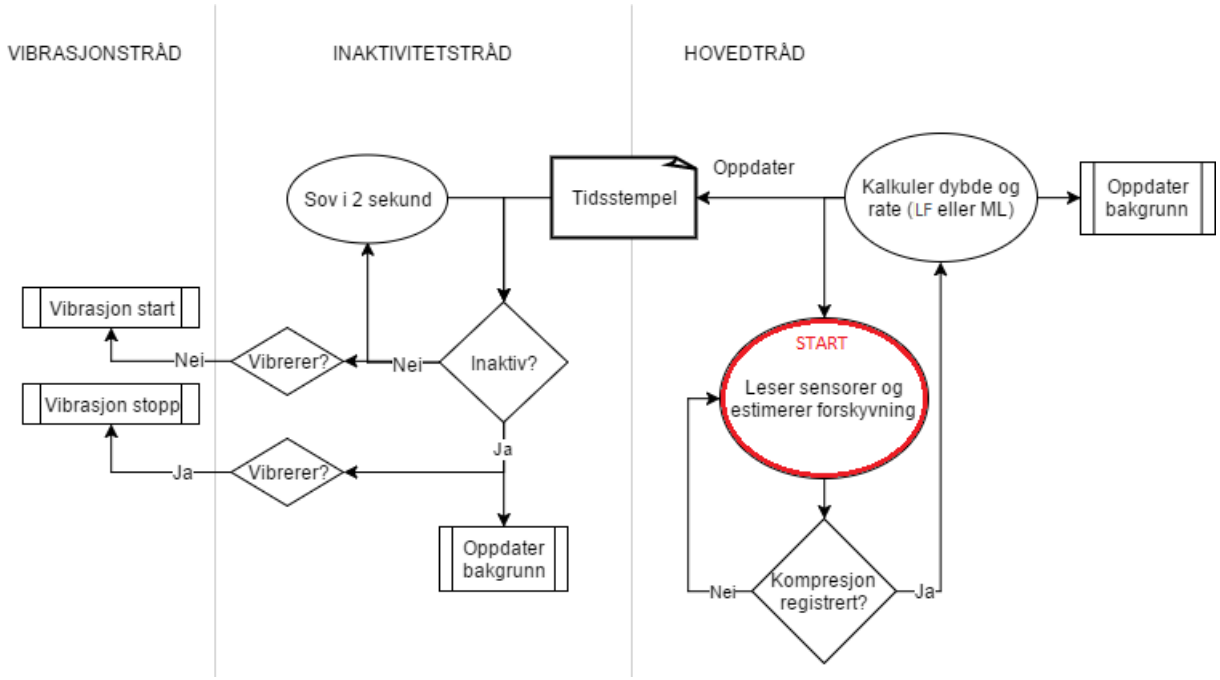
2.1 Android applikasjonen

I dette avsnittet beskrives appen som ble utviklet. Appen er minimalistisk utformet med kun få, men viktige, funksjoner. Når appen er startet

- ber den brukeren om å komprimere
- gir den tilbakemeldinger på hvor godt det komprimeres
- gir den beskjed når det er på tide med innblåsninger

Ved hjelp av innebygde sensorer kan appen beregne dybde og rate i HLR-kompresjoner. Etter kalkulasjoner på smartklokken er utført blir resultatet presentert i form av tilbakemeldinger til brukeren. Tilbakemeldingene opplyser kontinuerlig om kvaliteten på kompresjonene, og gis både visuelt på skjermen og fysisk via vibrasjoner. Uavhengig av LF- eller ML-metode kjører appen som vist i figur 6. Det er tre tråder, en hovedtråd, en inaktivitetstråd og en vibrasjonstråd.

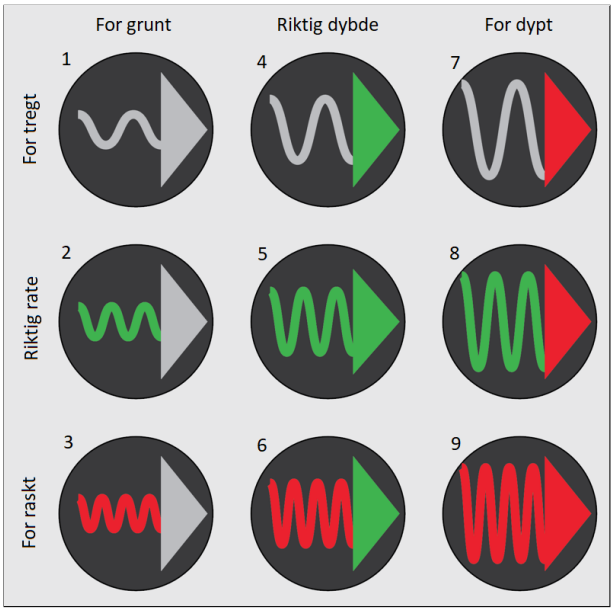
- Hovedtråden leser sensorene og beregner kompresjonsdybde og kompresjonsrate.
- Inaktivitetstråden sjekker om det aktivt komprimeres.
- Vibrasjonstråden sørger for vibrasjoner i riktig takt for kompresjonsrate.



Figur 6: Oversikt for appen

2.1.1 Design av app

Etter kompresjonsdybden og kompresjonsraten er kalkulert må resultatene presenteres for brukeren. Appen skal veilede brukeren i trening eller ved en virkelig hendelse, uansett en hektisk situasjon. Designet er derfor minimalistisk og bestående av to komponenter. Raten vises ved fargen på bølgen, og dybden vises ved fargen på pilen og ved høyden på bølgen. Designet er vist i figur 7. Intuitive farger som grå og rød signaliserer kompresjoner som ikke tilfredsstiller kravene. Grønt indikerer korrekte kompresjoner. Designet ble utviklet gjennom flere iterasjoner, beskrevet ytterligere i kapittel 4.1.



Figur 7: Design

På skjermen blir den oppnådde kompresjonsdybde og kompresjonsrate signalisert til brukeren. Skjermen oppdateres kun ved hver tredje kompresjon for å holde fokuset på komprimeringen og ikke overøse brukeren med informasjon. Dersom brystkompresjoner ikke er igangsatt, eller det er opphold i kompresjonene, vil klokken vise startskjermen i figur 8a. Appen er da i inaktiv modus. Appen kan være i to forskjellige modus, inaktiv og aktiv. Når en kompresjon er fullført, blir nåværende tidspunkt lagret. Inaktivitetstråden, vist i figur 6, sjekker annenhvert sekund dette tidspunktet opp mot nåværende tidspunkt for å se hvor lenge det er siden forrige kompresjon. Dersom det er to sekund eller mer går appen inn i inaktiv modus. I aktiv modus, hvis det er registrert 30 kompresjoner, endres bakgrunnen til å vise at det er på tide med innblåsning, vist i figur 8b. Det gis fire sekunder til rådighet, to sekund per innblåsning.

(a) Utfør kompresjoner

(b) Utfør innblåsninger

Figur 8: Ikoner for (a) kompresjon og (b) innblåsning

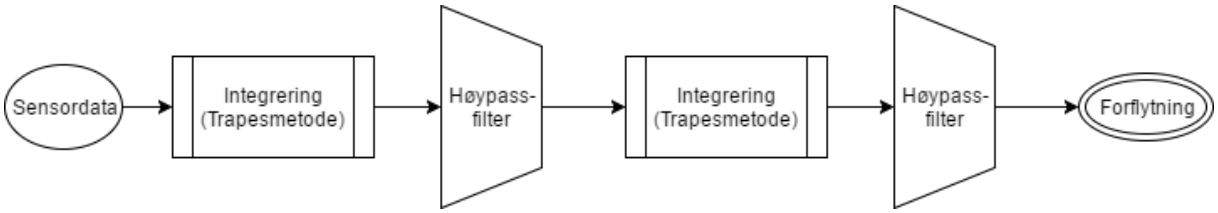
Ved å bruke gravitasjonen kan det avgjøres hvilken hånd klokken henger på. Ikonene i figur 7 og 8 viser ikonenes retning dersom klokken benyttes på venstre hånd. Disse blir horisontalt speilvendt om brukeren har klokken på høyre hånd.

2.1.2 Veiledning på kompresjonsrate

Klokken vibrerer med en takt på 110 per minutt, som styres av en egen vibrasjonstråd. Dette er den anbefalte kompresjonsraten (100-120 kpm) som skal veilede brukeren. Smartklokken benyttet i oppgaven har ikke innebygget høytaler, men andre klokker kan ha en metronom som piper til takten. Denne funksjonen er ikke implementert, men kan gjøres ved å benytte Android sin MediaPlayer klasse. Klasseobjektet laster inn en lydfil som kan startes med et start() funksjonskall, som kan utføres i vibrasjonstråden. Ved å implementere dette blir veiledningen både fysisk og audiovisuell.

2.2 Metode 1 - Lineær filtrering

I dette avsnittet beskrives den implementerte metoden for lineær filtrering. Metoden integrerer og filtrerer akslerasjonsdata for å beregne kompresjonsdybde. Som beskrevet i kapittel 1.6 er det mange små variasjoner i sensoren som slår ut i akselerasjonsmålingene. Ved å filtrere signalene kan de uønskede komponentene i signalet fjernes.



Figur 9: Integrering og filtrering av akselerometerdataene

2.2.1 Filtrering

Å velge riktig samplingsrate er en form for filtrering. Raten velges slik at det samples

- så raskt som mulig for maksimal nøyaktighet
- så tregt som mulig for å redusere prosessorbruk
- tregt nok for å unngå at støy dominerer signalet
- med en rate som holder seg stabil dersom prosessoren er under høyt forbruk

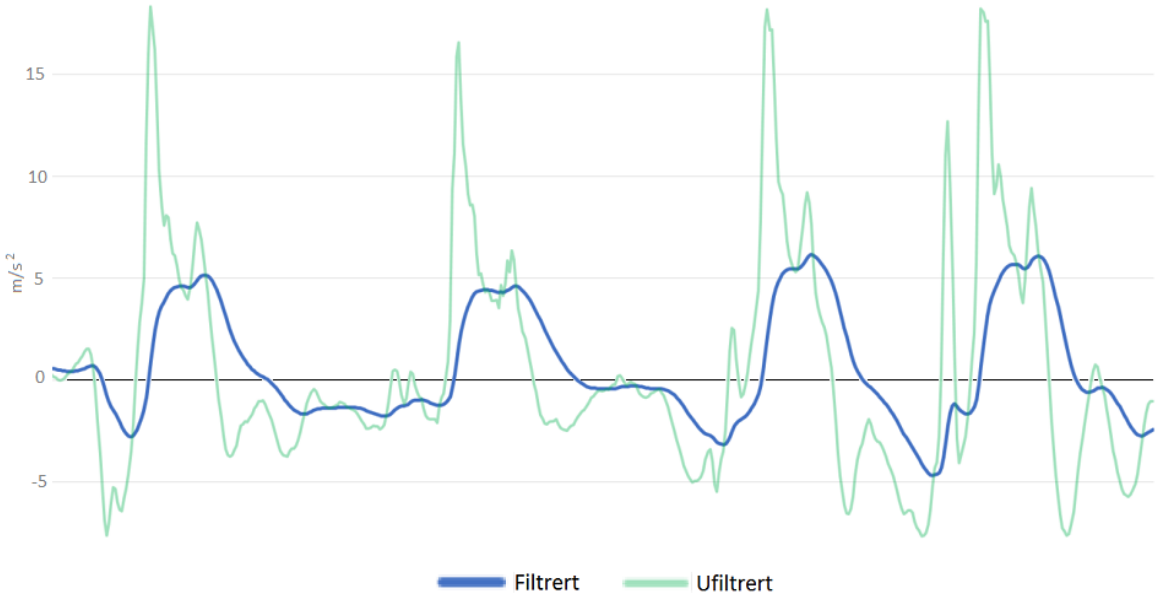
Smartklokkens raskeste samplingsrate blir benyttet. Ved denne samplingsraten fanges granulariteten i signalet opp med god margin. Med denne raten inneholder signalet en god del støy dersom klokken ligger i ro. Klokken skal derimot måle kompresjoner med relativt hard akselerasjon som vist i grønt i figur 10. Disse dataene er relativt støyfrie fordi akselerasjon fra

kompresjonene er dominerende. Ved å programmere klokken til å lese med denne hastigheten, er det kun et hint til operativsystemet i stedet for en kommando. Samplingsraten kan derfor varierer etter prosessorbruk. Ved utvikling av appen kjørte andre, lite ressurskrevende apper samtidig, og 200 Hz ble opprettholdt.

Lavpassfilteret vist i likning (5) lar kun langvarige endringer i signalet passere; det filtrerer ut kortvarige fluktuasjoner i signalet (støy). Koeffisienten β er satt til 0.95 som betyr at forrige måling, x_{n-1}, blir vektet til 95 prosent og den nye målingen, x_n, vektet til 5 prosent.

$$y_n = \beta \cdot x_{n-1} + (1 - \beta) \cdot x_n \tag{5}$$

Dette filteret benyttes kun i forbindelse med detektering av kompresjoner, ikke ved beregning av kompresjonsdybde. Ved å filtrere signalet kraftig, produseres en lett analyserbar kurve. Denne benyttes for å avgjøre når en kompresjon er fullført, og når en ny kompresjon starter. Figur 10 viser det filtrerte og ufiltrerte akselerasjonssignalet. Selv om det ufiltrerte signalet er relativt støyfritt, fluktuerer det en del rundt 0 og dermed er det filtrerte signalet enklere å analysere.



Figur 10: Lavpassfilter på akselerasjonssignal over fire manuelle kompresjoner

Høypassfilter vist i likning (6) lar kun kortvarige endringer i signalet passere. Når hastighet eller dybde er integrert, blir de tregtvarierende komponentene filtrert ut (drift). Vist i figur 9 anvendes filteret etter at de rå akselerometerdataene er integrert én gang til hastighet ved likning (1). Resultatet er filtrerte hastighetsdata som igjen kan integreres til dybde. I likningen er

$$y_n = \alpha \cdot (y_{n-1} + x_n - x_{n-1}) \tag{6}$$

- y_n er nye filtrerte måling
- y_{n-1} er forrige filtrerte måling
- x_n er nye ufiltrerte målingen
- x_{n-1} er forrige ufiltrerte målingen

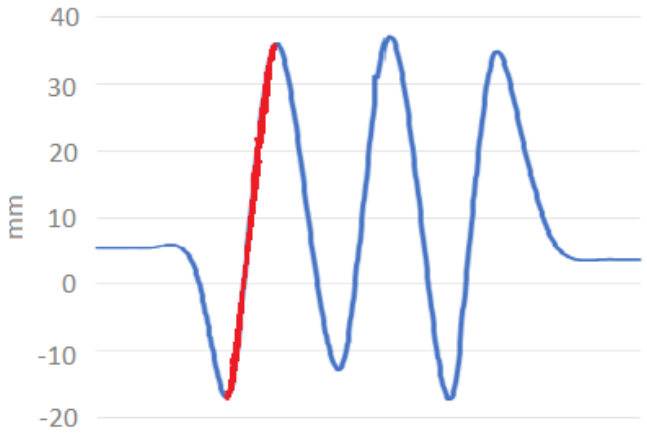
Ved filtrering av hastighet har driften en frekvensrespons under 1.50 Hz og koeffisienten α settes til 0.9550. Ved filtrering av dybde har driften en frekvensrespons under 0.50 Hz, og α satt til 0.9842.

2.2.2 Beregning av kompresjonsdybde og kompresjonsrate

Når det detekteres påfølgende målinger med negativ akselerasjon, vist i figur 10, er det muligens en kompresjon som er fullført. Det settes et flagg for å signalisere at algoritmen må undersøke om det virkelig var en kompresjon. I tillegg til flagget, er det to andre kriterier som må oppfylles før kompresjonsdybde og kompresjonsrate beregnes.

1. Påfølgende negativ akselerasjon (flagg satt).
2. Positiv akselerasjon på minimum 0,5 g (≈ 5 m/s^2) i løpet av kompresjonen.
3. Mindre enn ett sekund siden forrige kompresjon.

Dersom alle kriteriene er oppfylt, aksepteres kompresjonen. Et tidsstempel oppdateres ved hver kompresjon og dette blir satt i denne delen av algoritmen, uansett om kompresjonen aksepteres eller ei. På grunn av dette kriteriet, vil den første kompresjonen i en kompresjonssekvens falle fra da den ikke oppfyller kriterium 3. Metoden er implementert slik for å unngå falske kompresjonsmålinger når klokken beveges utenom komprimering. Kompresjonsdybde blir så beregnet med integreringsmetoden og filtrert med høypassfilteret. Den totale dybden, markert i rødt i figur 11, kalkuleres ved å subtrahere bunnpunkt fra toppunkt, som i dette tilfellet er $37mm - (-16mm) = 53mm$.



Figur 11: Beregnet forskyvning for tre kompresjoner

Kompresjonsraten beregnes etter hver kompresjon ved å ta et gjennomsnitt av tiden brukt på de fire siste kompresjonene, vist ved likning (7). Gjennomsnittet er rullerende; etter hver kompresjon blir ett nytt tidsstempel, t_n, lagt inn som overskriver det eldste tidsstempelet t_{n-4}.

$$Rate = \frac{60}{t_n + t_{n-1} + ... + t_{n-4}} \tag{7}$$

LF-metoden kunne benyttet gyroskopdata til å kompensere for vinkelendringer i kompresjonene. Dette ble ikke implementert og er diskutert i kapittel 4.4.2.

2.3 Metode 2 - Maskinlæringsalgoritme

For å implementere en ML-algoritme ble Google-utviklede TensorFlow benyttet da det har god støtte for operativsystemet Android. Med rå sensordata trenes algoritmen opp til å klassifisere ni ulike kategorier vist i tabell 2. Algoritmen ble utviklet i Python og deretter implementert på smartklokken.

Kategori	Dybde	Rate
1	Grunt	Tregt
2	Grunt	God
3	Grunt	Raskt
4	God	Tregt
5	God	God
6	God	Raskt
7	Dypt	Tregt
8	Dypt	God
9	Dypt	Raskt

Tabell 2: Testoppsett for ML-algoritmen

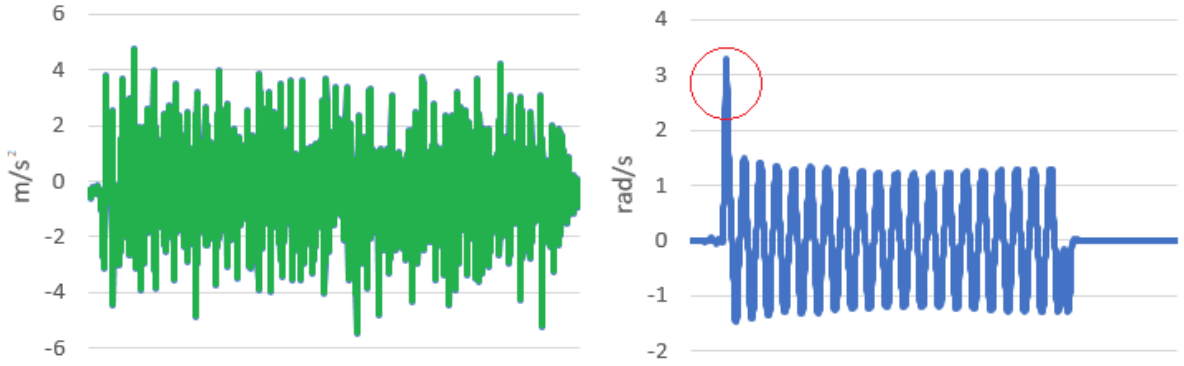
2.3.1 Konvolusjonelle nevrale nettverk

I denne metoden ble et KNN benyttet da de er godt egnet for den to-dimensjonale (2D) strukturen til sensordataene, akselerasjon vs. tid. Disse nettverkene er opprinnelig designet for bildegjenkjenning, der bilder konverteres til matriser av pixelverdier. Sensordata kan sees på samme måte, en 2D matrise av et kompresjonsbilde, der hver kompresjon er et unikt bilde. Et KNN inneholder ett eller flere konvolusjonslag, ofte med et nedsamplingssteg mellom konvolusjonslagene. Nedsampling brukes for å redusere datamengden uten å miste egenskapene til signalet.

2.3.2 Data til algoritmen

For å kunne klassifisere nye kompresjoner, må datasettet det trenes med inneholde varierte kompresjoner som representerer virkeligheten så godt som mulig. Dataene som skal brukes til trening og testing må være ulike. Dersom det testes med samme datasett som ble trent, vil det gi et mye bedre resultat enn når modellen eksponeres for ukjente data. Data ble samlet

inn ved bruk av industriroboten, beskrevet i kapittel 2.4. Før dataene kan trenes på algoritmen må de preprosseseres. Normalisering av data er nyttig dersom dataene har jevn variasjon. Ved å normalisere dataene blir alle egenskapene vektet likt i representasjonen. Data med uregelmessig variasjon, som toppen vist i figur 12b, påført normalisering vil sammentrykke de lavtvarierende dataene. Akselerometerdata normaliseres, men ikke gyroskopdata da de inneholder mange av disse toppene grunnet robotens kalibrerings-bevegelse i starten av hver kompresjonssesjon.



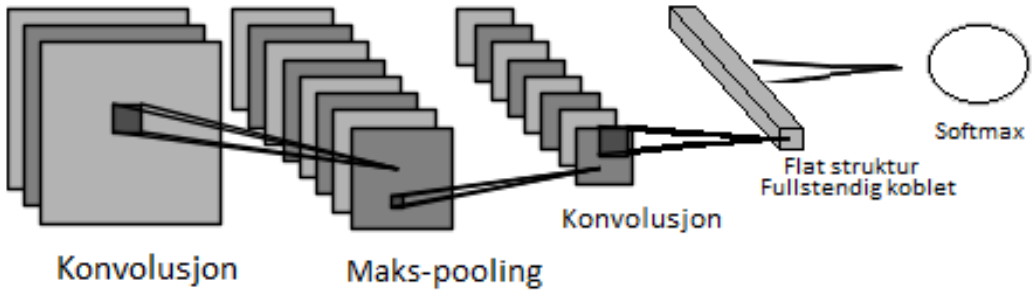
(a) Akselerometerdata

(b) Gyroskopdata

Figur 12: (a) Kan normaliseres (b) Kan ikke normaliseres

2.3.3 Oppbygging av algoritmen

Det er utviklet et KNN som er tilpasset sensordata fra brystkompresjoner. ML-algoritmen er et KNN som består av fem lag, vist i figur 13.



Figur 13: KNN

1. **Konvolusjon** - Kartlegge dataenes egenskaper. Det utføres multiplikasjoner mellom segmenter av inndataene og de egenskapene som algoritmen leter etter. Dersom egenskapen er tilstede vil multiplikasjonen produsere høye verdier som indikerer at egenskapen er funnet.

2. **Maks-Pooling** - Nedsampling av egenskapene funnet i lag 1. Reduserer totalt antall egenskaper i dataene men beholder de mest fremtredende.

3. **Konvolusjon** - Et nytt lag med kartlegging av egenskaper. De produserte dataene fra dette laget blir flatet ut før de ulike nevronene blir sammenkoblet.

4. **Fullstendig koblet** - Kalkulere inn vekter og bias i den flate strukturen til dataene. Resultatet er et fullstendig koblet nett.

5. **Softmax** - Nye inndata blir klassifisert til en av de ni kategoriene. I dette laget kalkuleres sannsynligheten for at dataene tilhører hver av de ni.

Det er ikke kjent at det eksisterer arbeid som benytter både akselerometer og gyroskop for å måle kompresjonsdybde, for hverken LF-metoden eller ved bruk av maskinlæring. ML-algoritmen er en godt dokumentert algoritme for å implementere et KNN [27].

2.3.4 Trening av algoritmen

Algoritmen ble trent med data fra eksperimentet som er beskrevet i kapittel 2.4. Inndataene til algoritmen ble merket med kategorier mellom 1 og 9 før de ble fôret til algoritmen. Hvert nevron definerer en operasjon som skal utføres på inndata og produsere utdata. Dersom nettverket ikke er eksponert for data og ikke inneholder noen vekter, vil ikke nettverket kalkulere noe. For å aktivere nettverket må det startes en TensorFlow sesjon som initialiserer alle variabler i nettverket. Når det fôres nye data vil operasjonene aktiveres og produsere utdata. Det er variablene, vekter og bias som definerer tilstanden til nettverket. Etter modellen er trent, lagres tilstanden for å kunne bruke modellen senere. I det fjerde laget kalkuleres vekter og bias inn. Modellen trenes med hele datasettet flere ganger for å lære dataene. Én iterasjon over settet kalles en epoke. For hver trente epoke er det disse verdiene som oppdateres, og når en tester data blir de bedre klassifisert etter en oppdatering.

Algoritmen har flere parametre som kan justeres etter behov. Algoritmen ble satt med start-parametre basert på [28], vist i tabell 3. Disse parametrene er tilpasset akselerometerdata logget med 20 Hz, men et større datasett. Antall epoker var satt til 5, noe som er altfor lite for å kartlegge egenskapene til dataene i denne oppgaven. Antall epoker ble satt til 2000. Det ble testet ut ulike kombinasjoner av parametrene for å avgjøre en bedre konfigurasjon for nettet. Algoritmen har definerte parametre og kan introduseres for data. Dataene som fôres er gruppert etter kategori. Alle gruppene blir fôret til nettet og en optimaliseringsfunksjon kjører etter hver epoke for å kalkulere vekter og tap. Tapet i modellen skal minimeres; lavere tap betyr bedre modell. Ved å utføre et større antall epoker kan det undersøkes hvordan modellen oppfører seg. Flere epoker er ikke alltid bedre; blir en modell trent for mye kan den pugge dataene og bli dårlig på å generalisere. Når modellen har lært dataene, har modellen konvergert. I tabell 3 vises startverdien for alle parametrene.

Inndata bredde er antall målinger som blir fôret om gangen. Antallet målinger blir delt opp i grupper bestemt av gruppestørrelsen.

Inndata høyde er satt til 1 da hver av de seks aksene med sensordata har én dimensjon, vist som høyde i figur 14.

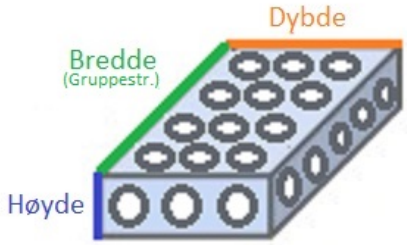
Parameter	Verdi
Inndata bredde	100
Inndata høyde	1
Gruppestørrelse	10
Kjernestørrelse	60
Filterdybde	60
Skjulte lag	1000
Læringsrate	0.0001
Epoker	2000

Tabell 3: Parametre for ML-algoritmen

Gruppestørrelse er et parameter som spesifiserer hvor store grupper inndata bredden deles opp i, vist som bredde i 14.

Kjernestørrelse er størrelsen på filteret som brukes ved nedsampling (maks-pooling). Et mindre filter bevarer flere egenskaper enn et større filter.

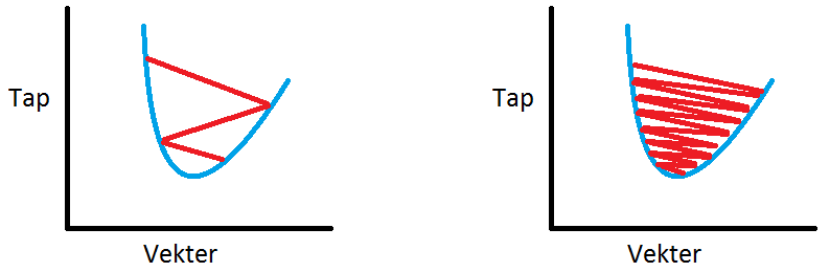
Filterdybde er dybden på hvor mange data filteret i konvolusjonslaget tar inn. Filterdybden er definert for hver akse individuelt, da filteret som benyttes er 2D-konvolusjon, vist som dybde i figur 14.



Figur 14: Høyde, bredde og dybde i konvolusjonsfilteret

Skjulte lag I figur 4 vises et nettverk med ett skjult lag med fire nevroner. Dersom det har to lag, vil alle nevronene i det første laget ha en kobling til nevroene i det andre laget. På disse koblingene er det en vekt. Flere skjulte lag vil gi flere koblinger og flere vekter som trenes, derav et større nett med bedre evne til å skille egenskapene det finner i inndataene.

Læringsrate er en parameter for hvor store steg algoritmen tar for å oppdatere vekter. Høy læringsrate betyr at stegene er større og modellen konvergerer raskere. Om stegene er for store, vil modellen ikke være nøyaktig nok til å oppnå den optimale vekten i forhold til tap, vist i figur 15a.



(a) Høy læringsrate - rask konvergering (b) Lav læringsrate - treg konvergering

Figur 15: Læringsrate

2.3.5 Testing av modell

For å se hvor godt nettverket yter, ble testsettet testet på modellen etter hver epoke, kun for å se hvordan modellen forbedrer eller forverrer seg. Dersom modellen trenes med testdata vil det gi for gode resultater når testdata valideres, og modellen vil klassifisere nye usette data dårligere.

Når algoritmen trenes til en modell, kalkuleres det et tap for hver epoke som sier hvor bra modellen klassifiserer. Tapet som kalkuleres er logaritmisk tap [29]. Tapet kvantifiserer nøyaktigheten ved å kalkulere høye tap for modellen når den klassifiserer feil. Ved å minimere tapet, maksimeres nøyaktigheten. Dersom modellen er veldig sikker på en kompresjon den har klassifisert, og det viser seg å være feil, vil det resultere i stort tap. Likning (8) viser logaritmisk tap der:

$$tap = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} log(p_{i,j}) \tag{8}$$

- N er antall observasjoner
- M er antall ulike kategorier
- log er naturlig logaritme
- $y_{i,j}$ er 1 dersom observasjonen er i riktig kategori, ellers 0
- $p_{i,j}$ er sannsynligheten for at observasjonen i tilhører kategorien j

2.3.6 Implementere TensorFlow på Android

Etter modellen er ferdig trent, må tilstanden fryses og lagres til fil, slik at den kan lastes inn igjen senere. For å laste inn modellen på smartklokken, må TensorFlow sitt programmeringsgrensesnitt for Android benyttes. Grensesnittet har blant annet funksjoner for å laste inn modellen, fóre den med nye data og hente ut resultater fra nevroner. Ved oppstart av appen lastes modellen inn i minnet og TensorFlow startes. Sensordata logges kontinuerlig. Når én eller flere kompresjoner er registrert, vil de oppsamlede sensordataene fóres inn til modellen via inngangsnevronet. Dataene klassifiseres og resultatet kan hentes ut fra endenevronet.

2.4 Eksperimentelle oppsett

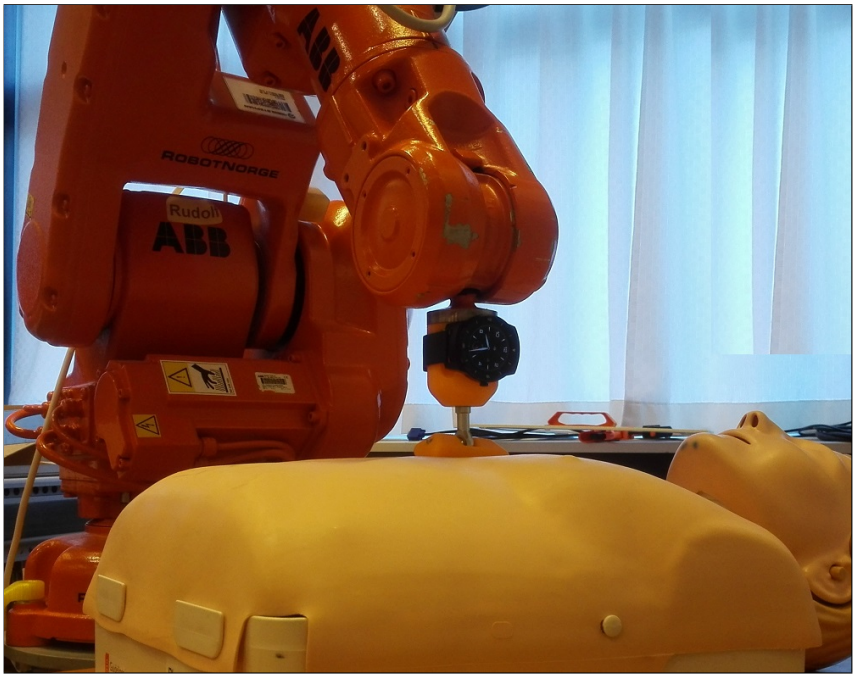
2.4.1 Automatisert testing ved bruk av industrirobot

Testingen har to formål:

1. Verifisere dybdemålingene fra smartklokkens LF-metode (2.2)
2. Innhente sensordata for å trene og teste ML-algoritmen (2.3)

For å utføre testing med industriroboten ble et håndledd 3D-printet og festet til enden av armen på roboten. Smartklokken ble så festet på håndleddet og dukken plassert under armen. Testoppsettet bestod av følgende utstyr vist i figur 16:

- Smartklokke
- ABB industrirobot
- 3D-printet håndledd
- Gjenopplivningsdukke



Figur 16: Eksperimentelt oppsett: Robot med håndledd, smartklokke og gjenopplivningsdukke

Roboten ble programmert til å utføre kompresjoner med ulike kombinasjoner av topp- og bunnvinkel for å simulere ekte kompresjoner. Vinklene ble funnet ved å ta video av en manuell kompresjonsesjon. Stillbilder ble hentet ut for topp- og bunnpunkt av en kompresjon og vinklene ble målt. Det ble funnet en vinkelforskjell på 5-10 grader mellom topp og bunn, som benyttes som grunnlag for vinklene i tabell 4. Alle sesjonene inneholder 20 kompresjoner for hver kombinasjon av dybde og rate. For hver unike rate, ble det utført sesjoner for alle dybdene. For å bestemme nøyaktigheten til dybdemålingene, ble smartklokken testet opp

mot en robot. Det ble utført et fastsatt antall sett med kompresjonssesjoner. Disse består av en kombinasjon av alle kompresjonsrater (60 - 140 kpm) og kompresjonsdybder (30 - 80 mm).

Sesjon	Toppvinkel	Bunnvinkel	Dybder (mm)	Rater (kpm)
1	7°	2°	30, 35, 40, ... , 80	60, 70, 80, ... , 140
2	10°	10°	30, 35, 40, ... , 80	60, 70, 80, ... , 140
3	12°	7°	30, 35, 40, ... , 80	60, 70, 80, ... , 140
4	15°	10°	30, 35, 40, ... , 80	60, 70, 80, ... , 140
5	15°	15°	30, 35, 40, ... , 80	60, 70, 80, ... , 140
6	20°	10°	30, 35, 40, ... , 80	60, 70, 80, ... , 140
7	20°	15°	30, 35, 40, ... , 80	60, 70, 80, ... , 140

Tabell 4: Testsesjonene

Dukken har innebygd optisk sensor for å måle kompresjonsdybde. Sensordataene ble sendt via en USB-kabel til datamaskinen. Dukken skulle egentlig plasseres under robotarmen, men robotens sikkerhetsmodus slår inn ved 45 mm dybde grunnet mengden kraft som måtte til for å komprimere fjæren i dukken. Det ble forsøkt å fjerne fjæren fra dukken, som gjorde at roboten komprimerte dypere, men dukken traff bunn på 70 mm slik at roboten igjen møtte motstand og stoppet. Dukken ble derfor utelatt fra testingen da roboten utfører samme bevegelser uavhengig av om dukken er under eller ikke. Ulempen med dette var at sensordata fra dukken ikke ble innhentet for sammenligning.

Etter at data ble logget fra klokken ble de behandlet og fôret til ML-modellen som beskrevet i kapittel 2.3. Dataene ble filtrert, segmentert og merket med en kategori for hvilke kompresjoner det var. Deretter ble dataene trent på modellen med følgende parametre i tabell 5, der antall epoker var satt til 2000. Når modellen ble trent, ble nøyaktigheten for treningsdata og testdata logget underveis, samt modellens treningstap og testtap.

Test	Filterdybde	Gruppestørrelse	Kjernestørrelse	Læringsrate	Skjulte lag	Aks/Gyro
1	60	25	60	0.0001	1000	Ja/Nei
2	60	10,25	60	0.0001	1000	Ja/Nei
3	60	25	60	0.1 - 0.00001	1000	Ja/Nei
4	60	25	30,60	0.0001	1000	Ja/Nei
5	60	25	60	0.00001	1000	Ja/Ja
6	30,60	25	30	0.00001	1000	Ja/Ja
7	60	25	30	0.00001	1 - 300	Ja/Ja
8	30	100	30	0.00001	200	Ja/Ja

Tabell 5: Testoppsett for ML-algoritmen

2.4.2 Manuell testing ved bruk av dukke

I tillegg til å benytte en robot til å komprimere, ble det utført manuelle kompresjoner for å verifisere kompresjonsdybden. Kun smartklokken og dukken ble benyttet i dette testoppsettet, vist i figur 17. I eksperimentet ble klokken festet slik en klokke normalt festes på håndleddet. Ved kompresjoner henger klokken så å si i ro, som ble verifisert ved å ta video av manuelle kompresjoner og analysere dem visuelt. Det ble utført 150 kompresjoner med varierende dybde (36 mm - 72 mm). Dukkens optiske sensor ble benyttet som referanse for dybdemålingene.

Figur 17: Gjenopplivningsdukken komprimeres manuelt med smartklokken på håndleddet

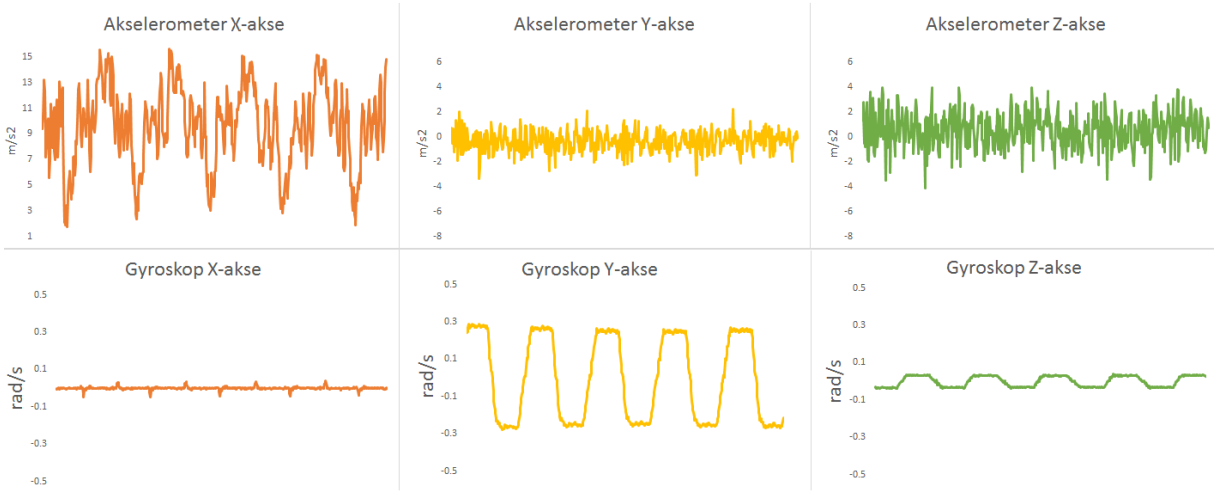
3 RESULTATER

3.1 Automatisert testing ved bruk av robot

I dette avsnittet presenteres resultatene fra eksperimentet i kapittel 2.4.1. Fra kompresjonssesjonene i tabell 4 ble det logget sensordata på smartklokken. Disse dataene utgjør datasettet i tabell 6. Roboten komprimerte totalt 13 860 kompresjoner i løpet av 4 timer. Et utsnitt av datasettet er vist i figur 18. Dataene fra smartklokken ble filtrert og nedsamplet, vist i tabell 6. De filtrerte dataene er påfølgende målinger som inneholdt kompresjoner, mens rådata inneholder 5 sekunders pause mellom sesjonene. Fra det filtrerte datasettet ble dybdedata hentet ut for sammenligning med robotens dybde i kompresjonene. Med det nedsamplede datasettet ble ML-modellen trent og testet, og beskrives i kapittel 3.3.

Datasett	Antall linjer	Størrelse	Samplingsrate	Antall kompresjoner
Rådata	2 350 000	124 MB	200 Hz	13 860
Filtrert	1 780 000	94 MB	200 Hz	13 860
Nedsamplet	356 000	19 MB	40 Hz	13 860

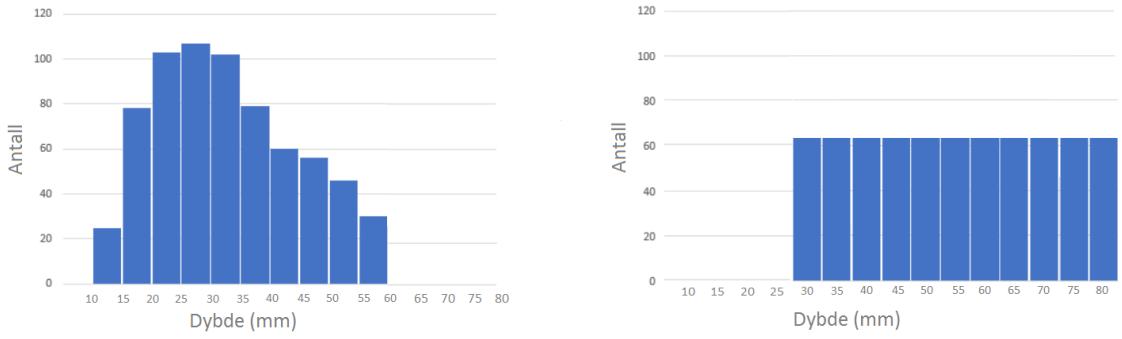
Tabell 6: Postprosessering på datasettet



Figur 18: Sensordata for gode kompresjoner (55 mm / 110 kpm) utført av robot

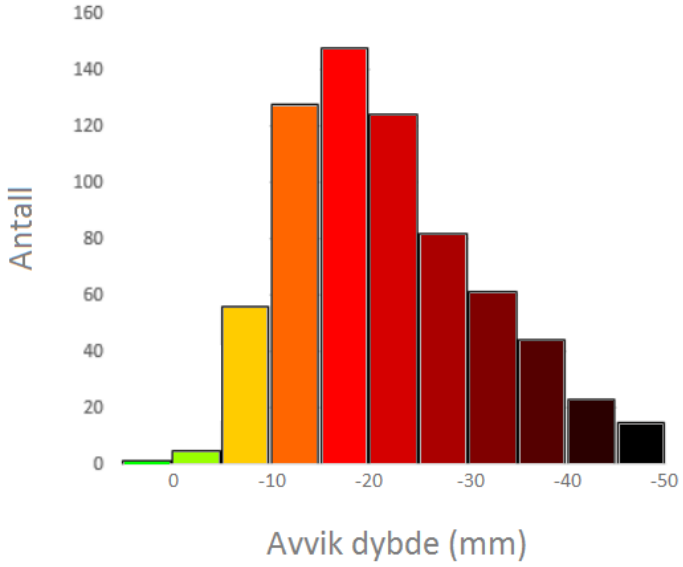
3.1.1 Presisjon og feilrater for kompresjonsdybde

For å verifisere kompresjonsdybden beregnet ved LF-metoden på smartklokken, ble målingene sammenlignet med målinger fra roboten. Figur 19 viser distribusjon av kompresjonsdybder fra klokken og fra roboten. Fra figur 19b er det tydelig at roboten har høy nøyaktighet for kompresjonsdybde og kompresjonsrate. Figur 20 viser avvikene på klokken i forhold til roboten der gjennomsnittlig avvik er på 21.8 mm, som er 40% av kompresjonsdybden ved gode kompresjoner. I kapittel 4.2 diskuteres årsaken til avvikene ytterligere; manuelle kompresjoner har en kraftigere akselerasjon som naturlig filtrerer dataene, og derfor ble de svakere robotkompresjonene mer påvirket av støy i signalet.



(a) Smartklokke (b) Robot

Figur 19: Distribusjon av kompresjonsdybde fra (a) smartklokkke og (b) robot



Figur 20: Avvik mellom LF-metoden og robot

3.1.2 Presisjon og feilrate for kompresjonsrate

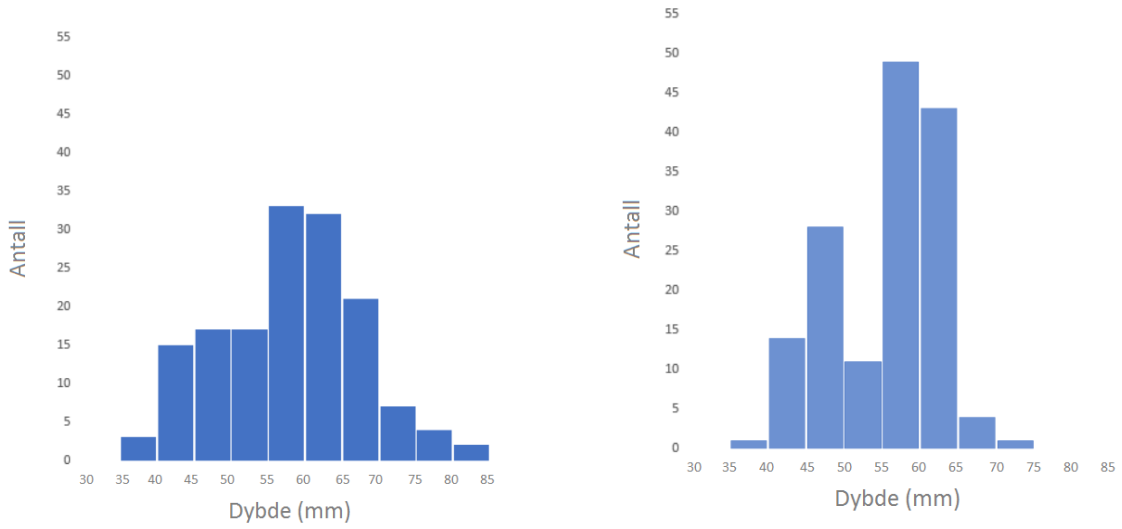
Kompresjonsraten beregnes ved tidsstempler, og ettersom algoritmen er lik uavhengig av automatiske robotkompresjoner eller manuelle kompresjoner, ble kun resultatene fra robottesten analysert. Kompresjonsraten beregnet på klokken, resulterte i et avvik i forhold til robotens kompresjonsrate, vist i tabell 7. Resultatet er beregnet fra 686 sesjoner med 20 kompresjoner i hver sesjon. Gjennomsnittlig avvik ble kun 0.63 kompresjoner per minutt, der 5.7 % av testene var avviket mer enn én kompresjon. Et avvik på 1 kompresjon kommer av metoden som benyttes for å registrere kompresjoner. Metoden godkjenner en kompresjon dersom det var mindre enn 1 sekund siden forrige kompresjon; en kompresjonssesjon på 20 kompresjoner blir da logget som 19 kompresjoner. Metoden registrerte derimot 49.3% korrekt. Dette kan forklares ved at kalibrerings-bevegelsen til roboten i starten av hver sesjon trigget algoritmen til å telle, og derfor endte algoritmen opp med å telle 1+19 kompresjoner.

Avvik	Antall	Andel
<-1	2	0.3 %
-1	5	0.7 %
0	338	49.3 %
1	304	44.3 %
2	22	3.2 %
>2	15	2.2 %
Alle	686	100 %

Tabell 7: Avvik i antall kompresjoner for alle sesjonene

3.2 Manuell testing ved bruk av dukke

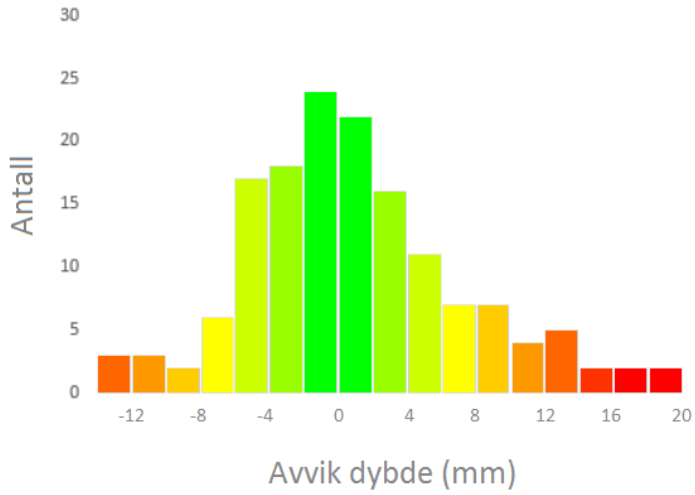
Det ble utført et eksperiment med manuelle kompresjoner gitt av én person, beskrevet i 2.4.2. Det ble utført 150 kompresjoner med varierende dybde (36 mm - 72 mm). Dukkens dybdemåling ble benyttet som referanse. Figur 21 viser gruppering av ulike kompresjonsdybder registrert på (a) klokken og på (b) dukken. Figur 22 viser avvikene for alle kompresjonene der gjennomsnittlig avvik er 3.2 mm.



(a) Smartklokke

(b) Dukke

Figur 21: Fordeling av kompresjoner registrert på (a) smartklokke og (b) dukke



Figur 22: Avvik per kompresjon gruppert etter avvik

3.3 Maskinlæring

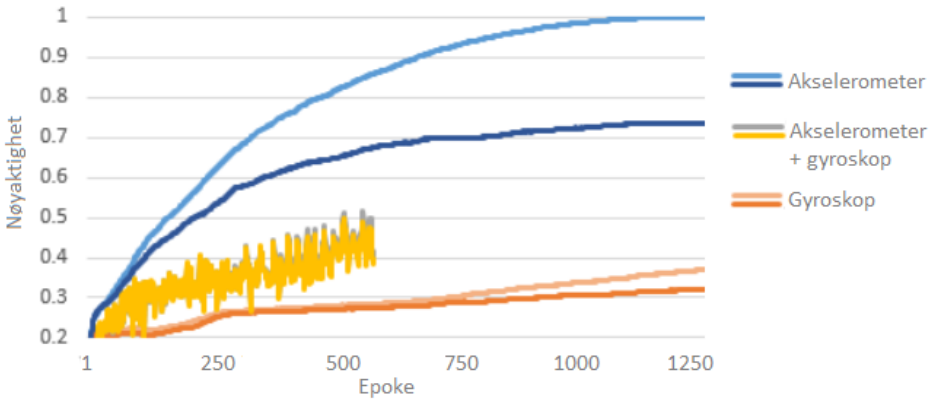
ML-algoritmen beskrevet i kapittel 2.3 ble fôret med datasettet og trent individuelt med hver av testene, vist i tabell 5. Hver test produserte resultater beskrevet i kommende avsnitt. Resultatene for alle testene er summert i tabell 8.

Test	Nøyaktighet	Epoker
1	73%	1100
2	76%	1700
3	79%	2400
4	79%	1500
5	84%	1700
6	72%	2000
7	88%	2700
8	90%	1100

Tabell 8: Resultatene viser nøyaktighet og antall utførte epoker når modellen har konvergert

3.3.1 Test 1 - Inndata 1

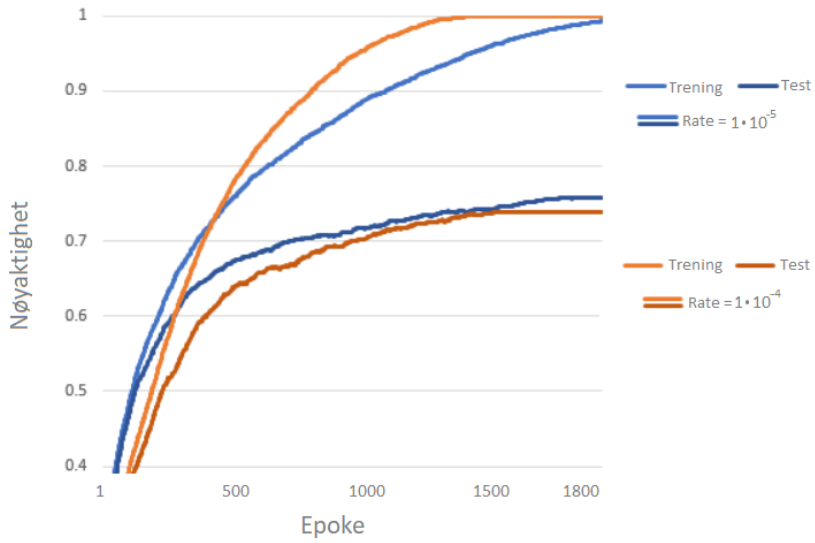
Ved å benytte alle dataene i datasettet, både akselerasjon- og gyroskopdata, ble resultatet som vist i figur 23. Hver for seg ga akselerasjon- og gyroskopdata brukbare resultater, men modellen konvergerte ikke ved å kombinere de. Den fluktuerende kurven kan forklares med at modellen gjettet mye underveis, selv om trenden på nøyaktighet økte. Årsaken til gjettingen var at z-aksen på gyroskopdataene var korrupt og algoritmen satt verdiene til 0 for alle z-akse gyroskopmålingene. Feilen ble rettet og er diskutert videre i kapittel 4.3.1. Test 2 og 3 ble derfor utført kun med akselerometerdata før feilen ble oppdaget.



Figur 23: Algoritmen ble fôret med ulike deler av datasettet

3.3.2 Test 2 - Læringsrate

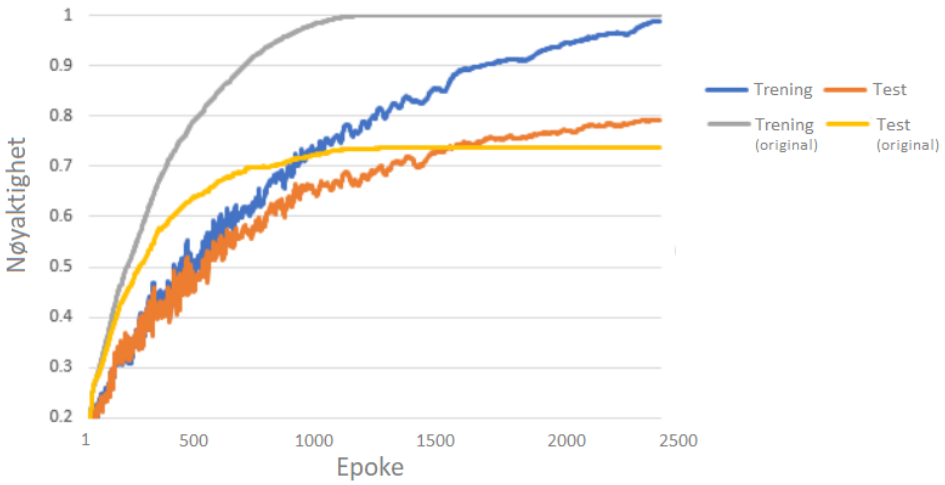
Det er nødvendig å undersøke ulike læringsrater for å finne de optimale vektene for nettverket. Testen ble utført med læringsratene $0.1, 0.01, 0.001, 0.0001$ og 0.00001. Kun de to sistnevnte produserte en brukbar modell, vist i figur 24. De tre første læringsratene fikk ikke modellen til å konvergere og stoppet på 18% nøyaktighet. Den laveste læringsraten 0.00001 oppnådde best resultat på 76%, mens 0.0001 oppnådde 73%.



Figur 24: Test av læringsrater

3.3.3 Test 3 - Kjernestørrelse

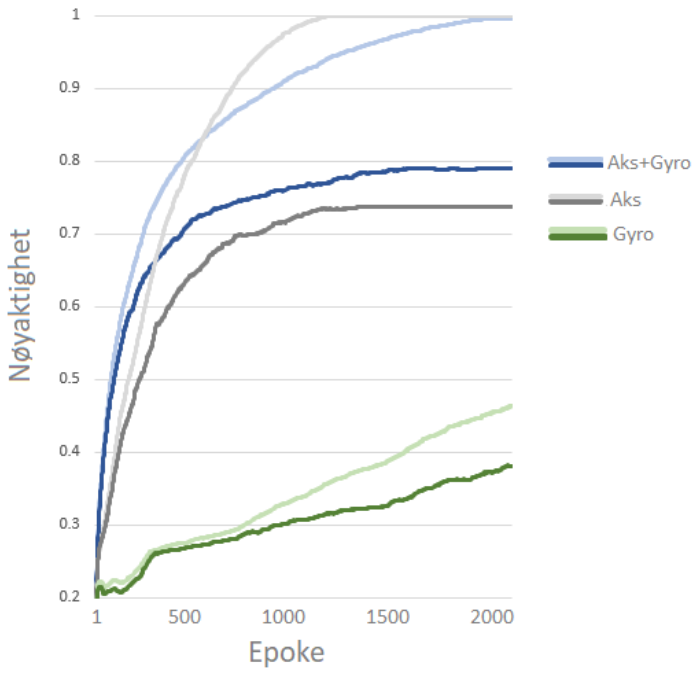
Ved å redusere kjernestørrelse konvergerer modellen over dataene tregere. Det ga flere nevroner totalt, og fler av dataenes egenskaper bevares i nedsamplingssteget til algoritmen. Resultater er vist i figur 25. Modellen oppnår en høyere nøyaktighet med kjernestørrelse 30 (79%) sammenlignet med original kjernestørrelse på 60 (74%).



Figur 25: Test med kjernestørrelser

3.3.4 Test 4 - Inndata 2

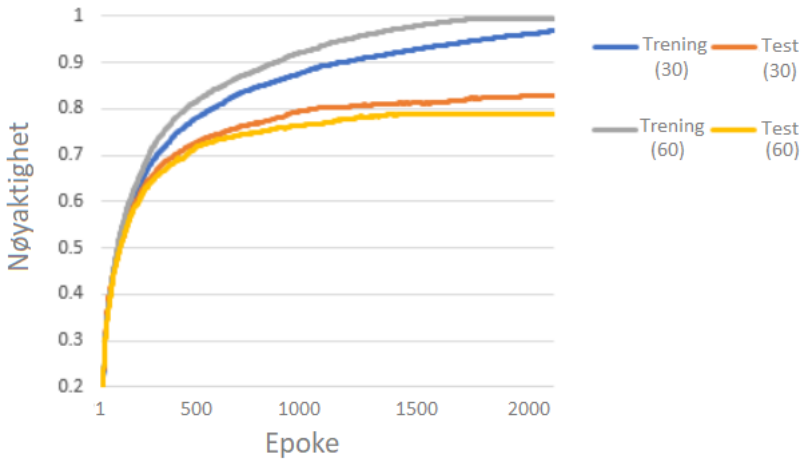
Da det ble oppdaget at de dårlige resultatene i Test 1 skyldtes en feil i preprosesseringen av dataene, ble en ny test utført på hele datasettet. Resultatet er vist i figur 26. **Kun gyroskop** som inndata gir en dårlig modell da vinkelendringene ikke varierer mye etterhvert som kompresjonsdybden endres. Modellen oppnådde i underkant av 40% nøyaktighet på gyroskopdata. **Kun akselerometer** som inndata resulterte i bedre nøyaktighet (73%), sammenlignet med gyroskopdata. **Både akselerometer og gyroskop** som inndata til algoritmen produserte de beste resultatene (79%).



Figur 26: Algoritmen fôret med ulike deler av datasettet

3.3.5 Test 5 - Filterdybde

Det ble først kjørt en test som implementerte parameterendringer fra Test 2-4. Dette økte nøyaktigheten fra 79% til 80%. Ved å endre på filterdybden, reduseres algoritmens mulighet til å pugge dataene, da det er færre nevroner i hvert konvolusjonelle lag. Færre nevroner gir et mer generaliserende nettverk. Filteret fikk halvert dybden fra 60 til 30 og fikk økt nøyaktigheten til 84%, vist i figur 27. Ved å redusere filterdybden ytterligere til henholdsvis 20 og 10, ble nøyaktigheten igjen redusert til 74% og 68%.



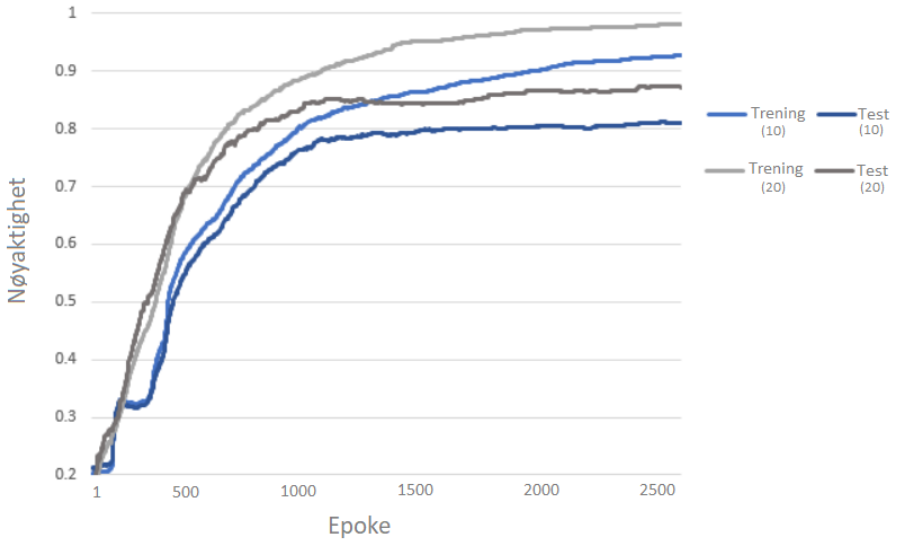
Figur 27: Dybdetest

3.3.6 Test 6 - Skjulte lag

Dype nettverk er bedre, men krever mer data. Ettersom modellen i de forrige testene har for lite data (diskuteres i 4.3.2), ble det forsøkt å redusere dybden på nettet. Det ble testet med 1, 2, 10, 20, 100, 200 og 300 skjulte lag. Det var ingen forbedring i nøyaktighet, men ved 10-200 lag konvergerte modellen bra, og oppnådde 72% for både treningsdata og testdata etter 500 epoker. Dette ble basis for valg av skjulte lag i den endelige modellen.

3.3.7 Test 7 - Gruppestørrelse

Gruppestørrelse er et parameter som deler opp antallet målinger, definert av bredden på inndata, i mindre grupper før det fóres til algoritmen. Denne testen ble kjørt med større gruppestørrelse på 100, i tillegg til færre skjulte lag (20 og 10). Ved å øke gruppestørrelsen til 100, blir det ved 40 Hz fóret cirka fire kompresjoner i stedet for én ved gruppestørrelse 25. Datasettet inneholder 20 påfølgende kompresjoner med lik topp- og bunnvinkel, kompresjonsdybde og kompresjonsrate. Dette gjør det mulig å benytte en gruppestørrelse på 100. Modellen oppnådde en nøyaktighet på 80% ved 10 skjulte lag og 88% ved 20, vist i figur 28.



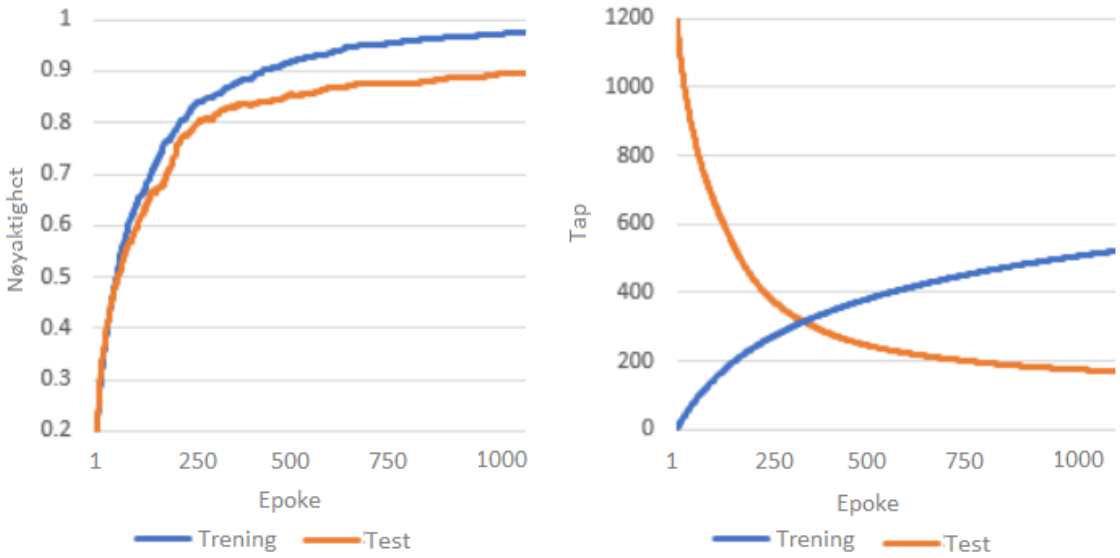
Figur 28: Breddetest

3.3.8 Test 8 - Optimaliserte parametre

Siste test ble utført med optimaliserte parametre i tabell 9 som baseres på forbedringer funnet i Test 1-7. Modellen konvergerte etter 1100 epoker, med en nøyaktighet på 90%. Gruppestørrelsen ble økt til 100. For å øke gruppestørrelse, må også bredden på inndata økes. I figur 29a vises den oppnådde nøyaktigheten. Modellens treningstap og testtap er kalkulert med likning 8 i kapittel 2.3.5.

Parameter	Verdi
Inndata bredde	200
Inndata høyde	1
Gruppestørrelse	100
Kjernestørrelse	30
Filterdybde	30
Skjulte lag	200
Læringsrate	0.00001
Epoker	1100

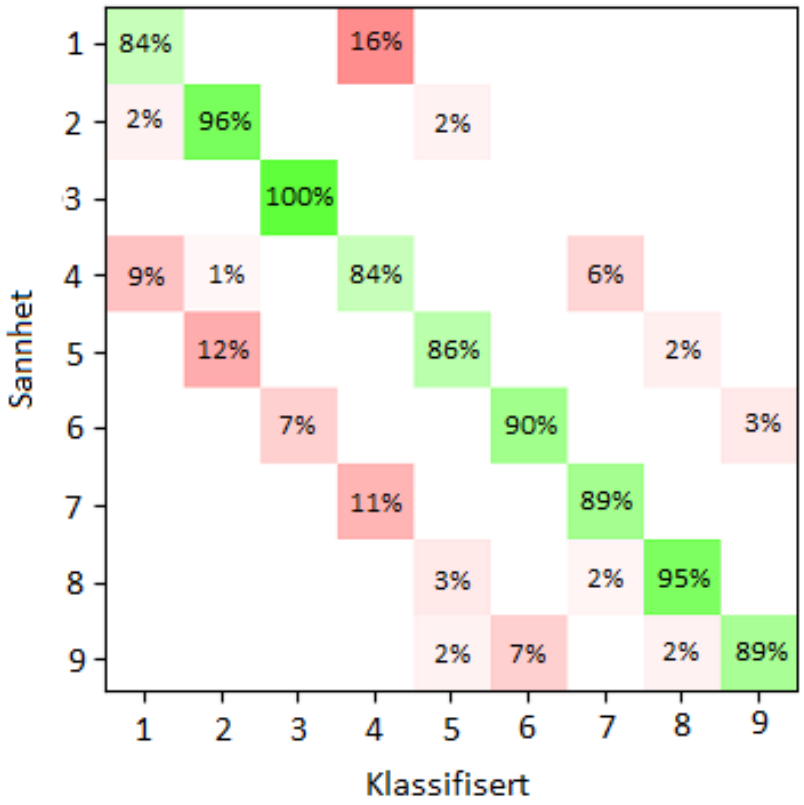
Tabell 9: Optimaliserte parametre for ML-algoritmen



(a) Nøyaktighet (b) Tap

Figur 29: Resultater fra Test 8

Modellen klassifiserte 90% av kompresjonene riktig. For å beskrive hvilke klassifiseringer som var korrekt benyttes en forvirringsmatrise (fra eng. confusion matrix). Klassifiseringene i Test 8 er vist i forvirringsmatrisen i figur 30. X-aksen viser hvilken kategori modellen klassifiserte kompresjonene til. De faktiske kategoriene langs y-aksen viser hvilken kategori klassifiseringene langs x-aksen tilhører. For kategori 1 vil det si at modellen klassifiserte 84% av kompresjonene til kategori 1 og 16% ble klassifisert til kategori 4.



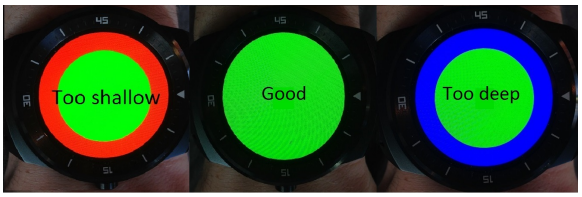
Figur 30: Forvirringsmatrise

4 DISKUSJON

I dette kapittelet diskuteres metodene som ble implementert i kapittel 2, begrensninger, alternative løsninger og forslag til forbedringer. Til slutt konkluderes oppgaven.

4.1 Applikasjonens design

I dette avsnittet diskuteres det endelige designet til appen samt hvordan designet utviklet seg gjennom flere iterasjoner. Appen har et minimalistisk design og kun ett formål: å veilede brukeren i sanntid til å utføre god HLR. Designet gikk gjennom to iterasjoner før det endelige designet ble valgt. Det første designet, vist i figur 31a, bestod av en sentrert sirkel for kompresjonsraten og en ring ytterst for kompresjonsdybden. Dette designet ble forkastet da de ni ulike kombinasjonene av fargene rød, grønn og blå ga en diskoaktig opplevelse. Det andre designet, vist i figur 31b, bestod av et enklere design som kun ga tilbakemelding på enten kompresjonsdybde eller kompresjonsrate, avhengig av hvilken som var mest kritisk å rette opp i. Dette designet ble forkastet da pilene som peker i ulik retning ikke er intuitivt. Det endelige designet er vist i figur 31c, der fargen på bølgen viser raten, og høyden på bølgen samt fargen på pilen viser dybden.



(a) Første design

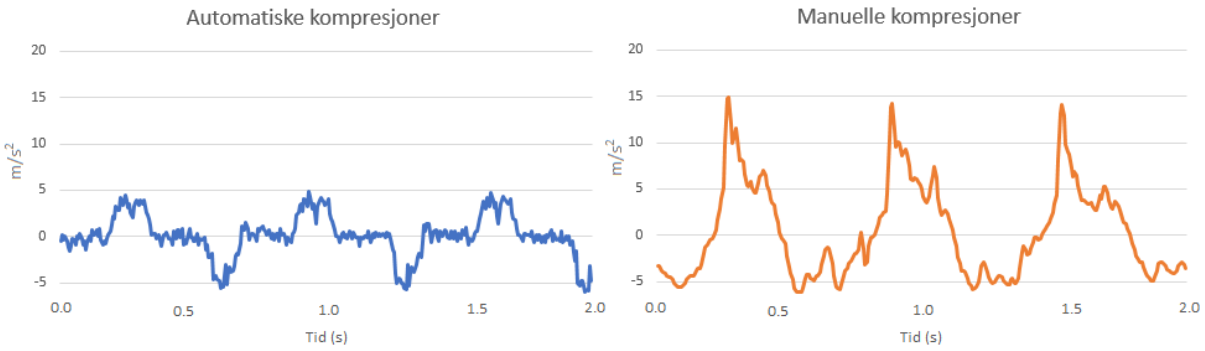
(b) Andre design

(c) Tredje design

Figur 31: Tre ulike design, viser tilbakemelding for grunne, gode og dype kompresjoner

4.2 Lineær filtrerings-metoden

Kun akselerometer ble benyttet i denne metoden. Det ga gode resultater ved manuelle kompresjoner på dukke. Resultatene for metoden varierte stort mellom manuelle kompresjoner og robotens kompresjoner. Ved manuelle kompresjoner ble signalet naturlig filtrert grunnet den bratte kurven vist i figur 32. Kompresjonene som roboten utførte hadde ikke en slik kurve og støy i signalet ble mer dominerende. Ved å filtrere sensordataene med et lavpassfilter før de integreres, kunne denne støyen blitt redusert. I stedet for å utføre roboteksperimentet på nytt, kunne lavpassfilteret blitt implementert i MATLAB og dybden beregnet på nytt for å verifisere metoden.



Figur 32: Akselerasjonsdata fra robot vs manuelle kompresjoner (60 mm / 100 kpm)

Kompresjonsdybden er beregnet kun fra akselerometerdata, som ikke tar hensyn til vinkelendringer i kompresjonen. Ved å implementere gyroskopmålingene i kalkulasjonene, kan det kompensere for vinkelendringene. Dette blir beskrevet ytterligere i kapittel 4.4.2. Det ble fokusert på å utvikle en ML-algoritme som kunne lære både gyroskop- og akselerometerdata i stedet for å implementere gyroskop i denne metoden.

4.2.1 Manuell testing ved dukke: Trusler mot validitet

Dybdeberegning med manuelle kompresjoner ga gode resultater med gjennomsnittlig avvik på 3.2 mm ved kompresjonsdybder fra 36 til 72 mm. Dette avviket utgjør maksimalt 9% ved 36 mm dybde. Dette kan bety at metoden gir tilstrekkelig tilbakemelding på kompresjonsdybde. Et tidligere arbeid beregner kompresjonsdybde med dobbelintegrasjon av akselerasjon og oppnår et gjennomsnittlig avvik på 2.3 mm [25].

Det er flere begrensinger ved denne type testing. Dukken som benyttes er et godt verktøy for HLR-trening, men forskjellig fra en virkelig pasient, både utførelsen av HLR i en reell situasjon og opplevelsen ved livredning vil være annerledes. Det er upraktisk og ikke minst uetisk å utføre en slik test på mennesker før det er utført en grundig studie på en dukke. Testen er begrenset til kompresjoner utført av kun én person. For å kunne avgjøre om metoden er nøyaktig nok må det utføres et utvidet eksperiment med en gruppe tilfeldige lekfolk Dette er gjort i arbeidet til Yeongtak Song et al. [25] der metoden ble vurdert som tilfredsstillende, med 2.3 mm gjennomsnittlig avvik og 0.9 mm standardavvik.

Når dybdeberegningene er basert på ett akselerometer er det kun ett referansepunkt i rommet. Metodene som ble utviklet avhenger derfor av at omgivelsene rundt pasienten er akselerasjonsfrie for å beregne nøyaktig. Om det utføres kompresjoner i et kjøretøy, vil ujevnheter og hull i veien samt svinger innvirke på akselerasjonen. Dette ble undersøkt i et arbeid der kompresjonsdybden ble estimert med kun ett akselerometer [21]. Det ble eksperimentert med å utføre HLR i en varebil (10 mm avvik) og i en åtte tonn tung båt på sjøen i 1.5 meters bølger (35 mm avvik). Om pasienten det utføres HLR på ligger på en madrass, vil komprimeringen av madrassen påvirke kompresjonsdybden og metodene vil underestimerer brystkassens kompresjonsdybde. Det anbefales å benytte hjertebrett for å redusere kompresjon av madrass, som utgjør cirka 8 mm på voksne pasienter i en alminnelig sykeseng [30]. Dersom hjertebrett ikke er tilgjengelig, kan kompresjonsdybden som beregnes, økes fra 50-60 mm til 60-70 mm [25].

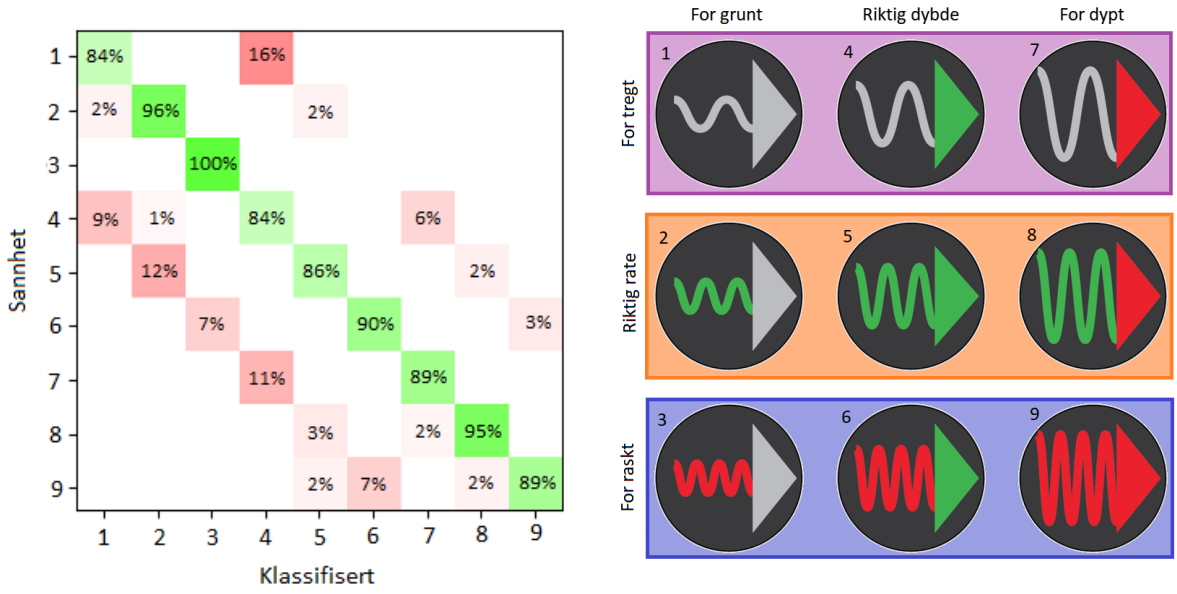
4.2.2 Beregning av kompresjonsrate

Metoden beskrevet i kapittel 2.2.2 består av å sjekke at kompresjonen har påfølgende negativ akselerasjon, positiv akselerasjon på minimum 0,5 g i løpet av kompresjonen og det er mindre enn ett sekund siden forrige kompresjon. Det ble registrert kompresjoner med rater på 60-140 kpm og med dybder på 30-80 mm, med et gjennomsnittlig avvik på kun 0.63 kompresjoner per minutt. Dette viser at metoden for å registrere at en kompresjon er fullført, gir et tilfredsstillende resultat.

4.3 Maskinlæring

Dette avsnittet vil diskutere ML-algoritmen som ble utviklet. Modellen klassifiserer med 90% nøyaktighet der algoritmen ble trent med 9702 kompresjoner og testet med 4158 kompresjoner. Resultatene viser at algoritmen kan klassifisere ulike dybder og rater godt, selv om vinkelendringene er store i løpet av kompresjonen, fra 2° til 20°. Modellen feilklassifiserte 416 kompresjoner, som kom frem i forvirringsmatrisen, gjentatt i figur 33a for lesbarhet . De grønne cellene er de korrekte klassifiserte kategoriene, som danner en diagonal linje. Denne linjen har to parallelle linjer, én til venstre og én til høyre. Begge linjene ligger 3 kategorier unna de korrekte klassifiseringene. Dette betyr at modellen feilklassifiserer kompresjoner med lik rate, men ulik dybde. Kategoriene som forveksles er gruppert i figur 33b. Dette er forventet, da kompresjonene er delt opp i kategoriene slik at kategorien for grunne kompresjoner går fra 30-45 mm, gode kompresjoner fra 50-60 mm og dype kompresjoner fra 65-80 mm. Det er mest sannsynlig ved dybdene 45-50 mm (og 60-65 mm) at modellen forveksler kategoriene og feilklassifiserer. For å undersøke dette, kan modellen testes med kun kompresjoner fra disse grenseverdiene. Dette ble ikke undersøkt nærmere, men er en mulig i videre arbeid.

Det er ingen fasit på hva som er akseptabel nøyaktighet for en ML-algoritme, det vil avhenge av bruksområdet og hvor nøyaktig eksisterende algoritmer er. Appen gir tilbakemelding på de tre siste kompresjonene og det er usikkert hvor mye feilklassifiseringene vil påvirke tilbakemeldingene. Dersom nye kompresjoner som skal klassifiseres er sentrert rundt en



(a) Forvirringsmatrise

(b) Kompresjonskategorier som forveksles

Figur 33: Forvekslede kategorier

spesifikk dybde, for eksempel 50 mm, er det sannsynlig å anta at den vil feilklassifisere noen av kompresjonene som grunne kompresjoner.

Da dataene ble samlet inn, var klokken festet på roboten tilsvarende om klokken var festet på høyre hånd. For LF-metoden er det uvesentlig da den benytter normalisert akselerasjon for alle aksene. For ML metoden, som tar inn akselerasjon og vinkelendringer som individuelle data, betyr dette at den mest sannsynlig ikke vil kunne klassifisere kompresjoner riktig om det komprimeres med klokken på venstre hånd. Datasettet bør derfor utvides med kompresjoner fra klokken på venstre hånd.

Resultatene viser at algoritmen kan klassifisere ulike kompresjonsdybder og kompresjonsrater godt, selv om det er betydelige vinkelendringer i løpet av kompresjonen. ML-algoritmen som er foreslått har forbedringspotensiale, og må trenes med data fra manuelle kompresjoner før den kan benyttes i HLR-trening, eventuelt at roboten komprimerer mer likt et menneske.

4.3.1 Feil i inndata til ML-algoritmen

Den første testen, presentert i kapittel 3.3.1, ga svært nedslående resultater på rundt 40% nøyaktighet, da den ble fôret med data fra både akselerometer og gyroskop. Det viste seg at gyroskopets z-akse manglet data, noe som ikke ble oppdaget før etter at den tredje testen av ML-algoritmen var utført. Årsaken til manglende data i z-aksen var en feil i kildekoden til Pythonskriptet som filtrerte dataene. Skriptet ekskluderte den sjette sensordata-kolonnen, nemlig gyroskopets z-akse. Dataene leses inn til ML-algoritmen fra et kommaseparert format, slik at manglende kolonne 6 fikk verdien til kolonne 1 i neste måling. Dermed ble rekkefølgen

stokket om og modellen klarte ikke å finne noe mønster i dataene. Test 1-3 ble ikke gjennomført på nytt grunnet liten tid da én test kan ta alt fra 8 til 48 timer på en laptop (i5-prosessor og 8GB RAM), avhengig av størrelsen på det nevrale nettet. Ved å utnytte GPUen til laptopen kan testtiden reduseres, TensorFlow har støtte for dette, men det ble ikke prioritert å implementere dette i oppgaven.

4.3.2 Overtilpasning på treningsdata

I resultatene for ML-algoritmen i kapittel 3.3 viser alle testene en treningsnøyaktighet betydelig høyere enn testnøyaktighet. Dette betyr at modellen overtilpasser dataene. Overtilpassing oppstår når algoritmen fanger opp støy i treningsdataene, og spesialiseres seg på å gjenkjenne disse dataene. Årsaken kan være en overdrevet komplisert model, der løsningen er å redusere kompleksiteten og benytte andre reguleringsmetoder for algoritmen, og deretter undersøke hvordan nøyaktigheten på testdata forbedrer seg. Dette ble utført i maskinlæringstestene i kapittel 3.3, der et grunnere nettverk med færre skjulte lag ble en bedre tilpasning til datasettet.

4.3.3 TensorFlow for Android

TensorFlow har god støtte for Android, men det er noe tungvint å finne gode guider på hvordan det skal implementeres. Google avslørte i mai 2017 at en optimalisert lettvekt versjon av TensorFlow, tilrettelagt for maskinlæring på håndholdte enheter, snart blir tilgjengelig[1]. En mindre ressurskrevende implementasjon er viktig for små systemer som smartklokker og smarttelefoner. TensorFlow modellen må fortsatt trenes på en datamaskin, da håndholdte enheter ikke har nok datakraft til å utføre beregningene innen rimelighetens grenser for prosesseringstid.

4.4 Forslag til forbedringer

4.4.1 Utvidet applikasjonen

For å forbedre brukeropplevelsen ville en smarttelefon koblet med appen være nyttig for å også få tilbakemeldingene på telefonen, som kan plasseres i bedre synsvinkel for brukeren. Når en app installeres på en smartklokke, er denne som oftest parret med en smarttelefon. Når appen kjører vil det være smartklokken som sender informasjon til smarttelefonen.

4.4.2 Implementasjon av gyroskop

Gyroskopet ble implementert i ML-algoritmen da det ga et bedre resultat enn kun akselerometer. Gyroskopet ble ikke implementert i LF-metoden, fordi akselerometeret alene ga tilstrekkelig presisjon ved manuelle kompresjoner. Ved betydelige vinkelendringer i kompresjonen vil ikke akselerometer alene være tilstrekkelig og LF-metoden kan forbedres ved å implementere gyroskopet.

[1] `https://techcrunch.com/2017/05/17/googles-tensorflow-lite/`

I et arbeid fra 2009 [23] ble det utviklet en metode for å beregne kompresjonsdybde i 3D ved bruk av ett akselerometer og to gyroskop. Metoden går ut på å måle vinkelen i starten av en kompresjon, måle vinkelendringene gjennom kompresjonen og til slutt kompensere kompresjonsdybden med topp- og bunnvinkel. Dersom det skulle implementeres vinkelkompensasjon i LF-metoden, ville det blitt løst på en tilsvarende måte som metoden til G. Zhang et al. [23].

4.4.3 Datasett med manuelle kompresjoner

Å trene ML-algoritmen på data fra manuelle kompresjoner, vil gi en modell som antakelig kan gjenkjenne virkelige kompresjoner mer nøyaktig enn ved å trene på data fra robotens kompresjoner. Som beskrevet i kapittel 4.2, er ikke robotkompresjonene like i karakteristikken på akselerasjon og gyroskopmålinger. Det er også en mulighet å programmere roboten til å utføre kraftigere kompresjoner med en mer ulineær akselerasjonskurve for å etterligne manuelle kompresjoner.

4.5 Konklusjon

Denne oppgaven presenterte to metoder for beregning av dybde i brystkompresjoner. Metodene ble implementert i en Android app for smartklokker. Det ble benyttet en gjenopplivningsdukke og en industrirobot for å evaluere presisjonen til metodene.

LF-metoden benyttet lineær filtrering av akselerasjondata til å beregne kompresjonsdybde. Dybden ble verifisert ved to eksperiment; ved manuelle kompresjoner på gjenopplivningsdukken og ved automatiske kompresjoner utført av industriroboten. Eksperimentene kan ikke verifisere kompresjonsdybdens nøyaktighet, men tilsvarende metode er verifisert i Yeongtak Song et al. [25], der metoden ble vurdert som tilfredsstillende. Denne løsningen begrenser seg til brystkompresjoner som utføres med små vinkelvariasjoner; implementasjon av et gyroskop kan kompensere for større vinkelendringer.

ML-metoden benyttet maskinlæring for å lære sensordata fra akselerometer og gyroskop. Datagrunnlaget var sensordata logget fra en smartklokke ved automatiske robotkompresjoner. Metoden ble verifisert til å klassifisere automatiske brystkompresjoner, utført av robot, med 90% nøyaktighet. Algoritmen som ligger til grunn for metoden må verifiseres ytterligere ved å lære og klassifisere flere data, før det kan konkluderes at nøyaktigheten er tilstrekkelig.

Resultatene viser at å benytte en smartklokke til å veilede en livredder er mulig med dagens smartklokker. Lineær filtrerings-metoden er en god metode, men nøyaktigheten kunne ikke verifiseres i denne oppgaven. Det antas at maskinlærings-metoden vil gi tilsvarende resultater for manuelle kompresjoner som automatiske kompresjoner og dermed benyttes i en smartklokke-app.

REFERANSER

[1] George Ritter mfl. «The effect of bystander CPR on survival of out-of-hospital cardiac arrest victims». I: *American Heart Journal* 110.5 (1985), s. 932–937. DOI: 10.1016/0002-8703(85)90187-5.

[2] Charles P. Friedman. «A "Fundamental Theorem" of Biomedical Informatics». I: *Journal of the American Medical Informatics Association* 16.2 (2009), s. 169–170. DOI: 10.1197/jamia.M3092.

[3] Voler Systems. *How to avoid big errors using accelerometers*. URL: http://volersystems.com/v-2011/125-how-to-avoid-big-errors-using-accelerometers/ (sjekket 25.01.2017).

[4] S. Ha og S. Choi. «Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors». I: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, s. 381–388. DOI: 10.1109/IJCNN.2016.7727224.

[5] Knut Jostein Knutsen. *Formelsamling i fysikk*. Tapir, 1970, s. 7.

[6] Lloyd N Trefethen og J A C Weideman. «The Exponentially Convergent Trapezoidal Rule». I: *SIAM Review* 56.3 (2014), s. 385–458. DOI: 10.1137/130932132.

[7] Folkehelseinstituttet. *Dødsårsaksregisteret: Dødsfall av hjerte-karsykdommer (D5)*. URL: http://statistikkbank.fhi.no/dar/ (sjekket 10.02.2017).

[8] Helsedirektoratet. *Nordmenn er gode på hjerte-lunge-redning*. URL: https://helsedirektoratet.no/nyheter/nordmenn-er-gode-pa-hjerte-lunge-redninghlr (sjekket 10.02.2017).

[9] Jasmeet Soar mfl. «European Resuscitation Council Guidelines for Resuscitation 2015». I: *Resuscitation* 95 (2015), s. 100–147. DOI: 10.1016/j.resuscitation.2015.07.016.

[10] Catherine J. Black, Anthony Busuttil og Colin Robertson. «Chest wall injuries following cardiopulmonary resuscitation». I: *Resuscitation* 63.3 (2004), s. 339–343. DOI: 10.1016/j.resuscitation.2004.07.005.

[11] Richard O. Cummins og Mary Fran Hazinski. «The most important changes in the international ECC and CPR Guidelines 2000». I: *Resuscitation* 46.1-3 (2000), s. 431–437. DOI: 10.1016/S0300-9572(00)00301-4.

[12] Michael R. Sayre mfl. «Hands-only (compression-only) cardiopulmonary resuscitation: A call to action for bystander response to adults who experience out-of-hospital sudden cardiac arrest - A science advisory for the public from the American heart association emergency cardiovascular

care committee». I: *Circulation* 117.16 (2008), s. 2162–2167. DOI: 10.1161/CIRCULATIONAHA.107.189380.

[13] Wissenberg M mfl. «Association of national initiatives to improve cardiac arrest management with rates of bystander intervention and patient survival after out-of-hospital cardiac arrest». I: 310.13 (2013), s. 1377–84. DOI: 10.1001/jama.2013.278483.

[14] Digna M. González-Otero mfl. «A New Method for Feedback on the Quality of Chest Compressions during Cardiopulmonary Resuscitation». I: *BioMed Research International* 2014 (2014), s. 21–23. DOI: 10.1155/2014/865967.

[15] Statista. *Smartwatch unit sales worldwide from 2014 to 2018 (in millions)*. URL: https://www.statista.com/statistics/538237/global-smartwatch-unit-sales/ (sjekket 11.02.2017).

[16] Y Jeong mfl. «Smartwatch app as the chest compression depth feedback device». I: *World Congress on Medical Physics and Biomedical Engineering, June 7-12, 2015, Toronto, Canada*. Bd. 51. Springer. 2015, s. 1465–1468. DOI: 10.1007/978-3-319-19387-8_357.

[17] Sofía Ruiz de Gauna mfl. «A Feasibility Study for Measuring Accurate Chest Compression Depth and Rate on Soft Surfaces Using Two Accelerometers and Spectral Analysis». I: *BioMed Research International* 2016, 7 pages (2016). DOI: doi:10.1155/2016/6596040.

[18] Gohier F, Dellimore KH og Scheffer C. «Development of a real-time feedback algorithm for chest compression during CPR without assuming full chest decompression». I: *Resuscitation* 85.6 (2014), s. 820–825. DOI: 10.1016/j.resuscitation.2014.03.003.

[19] Sofía Ruiz de Gauna mfl. «Feedback on the Rate and Depth of Chest Compressions during Cardiopulmonary Resuscitation Using Only Accelerometers». I: *PLOS ONE* 11.3 (mar. 2016), s. 1–17. DOI: 10.1371/journal.pone.0150139.

[20] Yeongtak Song mfl. «Development of Android Based Chest Compression Feedback Application Using the Accelerometer in Smartphone ». I: *Proceedings of the International Conference on Biomedical Engineering and Systems*. Avestia, 2014, Paper No. 130. URL: http://avestia.com/ICBES2014_Proceedings/papers/130.pdf (sjekket 26.01.2017).

[21] Sven Ole Aase og Helge Myklebust. «Compression depth estimation for CPR quality assessment using DSP on accelerometer signals». I: *IEEE Transactions on Biomedical Engineering* 49.3 (2002), s. 263–268. DOI: 10.1109/10.983461.

[22] Kjersti Engan mfl. «Chest compression rate measurement from smartphone video». I: *BioMedical Engineering OnLine* 15.1 (2016), s. 95. DOI: 10.1186/s12938-016-0218-6.

[23] G. Zhang, J. Zheng og T. Wu. «A Novel Method of Measuring the Depth of Manual Chest Compressions During CPR». I: *2009 3rd International Conference on Bioinformatics and Biomedical Engineering*. Jun. 2009, s. 1–4. DOI: 10.1109/ICBBE.2009.5162221.

[24] Tomohiko Sakai mfl. «Cardiopulmonary Resuscitation Support Application on a Smartphone–Randomized Controlled Trial –». I: *Circulation Journal* 79.5 (2015), s. 1052–1057. DOI: 10.1253/circj.CJ-14-1258.

[25] Yeongtak Song mfl. «Smartwatches as chest compression feedback devices: A feasibility study». I: *Resuscitation* 103 (2016), s. 20–23. DOI: 10.1016/j.resuscitation.2016.03.014.

[26] Peter A. Meaney mfl. «Cardiopulmonary Resuscitation Quality: Improving Cardiac Resuscitation Outcomes Both Inside and Outside the Hospital». I: *Circulation* 128.4 (2013), s. 417–435. DOI: 10.1161/CIR.0b013e31829d8654.

[27] Song-Mi Lee, Sang Min Yoon og Heeryon Cho. «Human activity recognition from accelerometer data using Convolutional Neural Network». I: *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. Feb. 2017, s. 131–134. DOI: 10.1109/BIGCOMP.2017.7881728.

[28] Aaqib Saeed. *Implementing a CNN for Human Activity Recognition in Tensorflow*. URL: http://aqibsaeed.github.io/2016-11-04-human-activity-recognition-cnn/ (sjekket 03.04.2017).

[29] Léon Bottou. «Large-Scale Machine Learning with Stochastic Gradient Descent». I: *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Physica-Verlag HD, 2010, s. 177–186. DOI: 10.1007/978-3-7908-2604-3_16.

[30] Akira Nishisaki mfl. «Backboards are important when chest compressions are provided on a soft mattress». I: *Resuscitation* 83.8 (2012), s. 1013–1020. DOI: 10.1016/j.resuscitation.2012.01.016.

[31] Chiwon Ahn mfl. «Evaluation of Smartphone Applications for Cardiopulmonary Resuscitation Training in South Korea». I: *BioMed Research International* 2016 (2016). DOI: 10.1155/2016/6418710.

[32] Malta Hansen C mfl. «Association of bystander and first-responder intervention with survival after out-of-hospital cardiac arrest in north carolina, 2010-2013». I: *JAMA* 314.3 (2015), s. 255–264. DOI: 10.1001/jama.2015.7938.

[33] Agnes Gruenerbl mfl. «Smart-watch Life Saver: Smart-watch Interactive-feedback System for Improving Bystander CPR». I: *Proceedings of the 2015 ACM International Symposium on Wearable Computers*. ISWC '15. ACM, 2015, s. 19–26. DOI: 10.1145/2802083.2802086.

[34] Andrew Ng mfl. *Unsupervised Feature Learning And Deep Learning Tutorial*. URL: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/ (sjekket 15.04.2017).