Forord

Oppgavens hensikt var å evaluere tidligere arbeid innenfor klassifisering av øyedata i virtuell virkelighet, og fortsette dette arbeidet ved å videreutvikle eksisterende- og implementere nye løsninger. Oppdragsgiver ønsket å utforske til hvilken grad det var mulig å benytte seg av maskinlæring til å klassifisere og predikere øyebevegelser.

Gruppen så på dette som et utmerket anledning til å benytte og videreutvikle egen kunnskap om maskinlæring. Samtidig ville dette være gruppens først mulighet til å utforske hele forskningsprosessen. Oppgaven stilte krav til innhenting, evaluering og anvendelse av data samlet fra øyet. Å få være med på en prosess som leder til økt kunnskap om et så nytt domenet som bruk av maskinlæring på øyedata, var for gruppen en stor motivasjonsfaktor.

Prosjektet har til tider vært utfordrende, men til enhver tid spennende og givende. Gruppemedlemmene har fått anvendt kunnskap som de har opparbeidet seg gjennom tre år på studiet, samtidig som de har tilegnet seg ny kunnskap og erfaringer. Gruppen er stolte over resultatene, og motiverte for å gripe fatt på nye oppgaver.

Gruppemedlemmene ønsker å uttrekke sine takksigelser til veileder og oppdragsgiver Alexander Holt, som gjennom prosjektarbeidet har bidratt med innsikt og konstruktive tilbakemeldinger. Holt har gitt gruppen spillerom til å styre retningen til prosjektet selv, og vist åpenhet og iver for å utforske det de selv mente var hensiktsmessig. Avslutningsvis takker gruppen Ali Alsam, samt Martin Moan, Marius Nygård og Håvard Ramberg som alle har bidratt med kunnskap og innsikt.

19.05.2022 - Sem Sælands vei 9, 7051 Trondheim

Simon Jensen | Stian Fjæran Mogen | Lars Brodin Østby

Oppgavetekst

Original oppgavetekst

Det har tidligere vært gjennomført en bacheloroppgave hvor det ble implementert forskjellige VR-miljø/eksperimentscener for å kunne forske på å bruke maskinlæring for å hurtig kunne avgjøre en persons blikk-fokus i VR. Vi vil nå videreføre denne oppgaven, og med utgangspunkt i det tidligere arbeidet gjort, er vi interessert i å samle inn mer data for å kunne utføre en grundigere evaluering av det tidligere arbeidet. I tillegg til dette er vi interessert se på brukbarheten av systemet med tanke på to nye aspekter: prediksjon av øyebevegelser (gitt øyets nylige bevegelser, kan vi forutsi med rimelig sannsynlighet hvor det kommer til å bevege seg et kort stykke inn i fremtiden?), og kognitiv oppmerksomhet/fokus (hva en person faktisk er oppmerksom på i det virtuelle miljøet).Oppgaven er av akademisk art og må sies å være forskningsrettet, men det vil likevel være nødvendig å kunne implementere ny funksjonalitet på det eksisterende systemet. Det er et mål at resultatet av oppgaven blir brukt i videre forskning, og vi har andre parallelle prosjekter som vil kunne dra nytte direkte nytt av arbeidet som vil bli gjort. Det anbefales at studentene som melder interesse for oppgaven har god kjennskap til maskinlæring og programmering.

Revisjon av oppgavetekst

Oppgaveteksten som presentert i oppstartsfasen har vært gruppens utgangspunkt for forskningen, og har blitt evaluert gjennom kontinuerlig dialog sammen med produkteier og veileder. Det faktum at nye funn og erfaringer i løpet av prosjektforløpet kunne endre fokusområder, er noe alle involverte parter var klar over. Likevel har oppgaveteksten holdt seg relativ lik slik den var i oppstartsfasen, med enkelte spesifikke endringer basert på vurderinger tatt av gruppen sammen med veileder. Oppmerksomhet er vanskelig å klassifisere med en form for fasit. Å identifisere visuell input fra øyedata er derimot en disiplin som både er håndfast og etterprøvbar. Det samme gjelder å identifisere perser basert på deres øyedata. Disse endringene skal gjøre oppgaven og de tilhørende resultatene mer håndfaste, samtidig som de på lik linje med den originale oppgaveteksten fasiliterer for videre forskning på området. Innsamling av mer data og vurdering av tidligere arbeid er fortsatt sentrale deler av oppgaveteksten, mens prediksjon av øyets bevegelser frem i tid blir sett på som potensiell videre utvikling.

Sammendrag

Eyetracking handler om å måle øyebevegelser for å finne ut hvor en person ser, hva de ser på og hvor lenge blikket er på et bestemt punkt. Ettersom øynene våre er et av de primære verktøyer vi bruker for beslutningstaking, kan eyetracking benyttes av forskere til å studere menneskelig atferd og forstå visuell oppmerksomhet. Dette bachelorprosjektet samt tilhørende rapport bygger på et tidligere bachelorprosjekt gjennomført av Moan, Nygård og Ramsberg i 2020. Det forgående prosjektet fokuserte på oppsett av scener i VR for innsamling av øyedata, for å så implementere en rekke algoritmer for best mulig klassifisering av denne dataen. Det nåværende prosjektet plukker på mange måter opp der det prosjektet slapp, samtidig som det tar forskningen i nye retninger. Prosessen har ledet til et skalerbart forskningsprosjekt, som er designet for videre utvikling. Resultatene som presenteres i denne rapporten foreslår at det er mulig å klassifisere både visuell input og brukere av VR-headsett, med maskinlæring trent på den tilhørende øyedataen. Denne dataen kan også benyttes til å predikere fremtidige sakkadiske bevegelser.

En av de store begrensninger i 2020, kom ved innsamling av tilstrekkelig variert data i koronapandemien. Dette var ikke en begrensende faktor ved denne prosjektgjennomføringen og tillot innsamling av et godt datagrunnlag. Gruppen utførte en nøye evaluering av det tidligere utførte arbeidet, samt konklusjoner som ble dratt av den forgående gruppen. Dette er blitt brukt til analysen av klassifiseringsalgorimer, og vist at det er store ulikheter i hvordan disse skiller på sakkader og fikseringer. Dette gjelder både for algoritmer seg imellom, og i sammenligning av ulike scener. Dette ble bakgrunnen for utviklingen av fire nye eksperimenter, som er designet for å fremprovosere forskjellige typer øyebevegelser som er representert i litteraturen. Disse representerer dens visuelle input.

Det er samlet inn over 300mb med øyedata for analysen. Ved hjelp av maskinlæring og data fra de nye scene utviklet gruppen modeller som skulle være i stand til å predikere hvilken scene data tilhørte. CNN-modellen ga et godt utgangspunkt, med en presisjon over 60 prosent for klassifisering av scener. De beste resultatene kom fra med RNN-modellen, som på uavhengig testsett klassifiserte scener og testpersoner med presisjon opp mot 90 prosent. RNN-modellene ble implementert i Unity for klassifisering i sanntid, her også ble resultatene gode, med noe lavere presisjon som følge av spesifikke implementasjonsdetaljer. Det var også av interesse å kunne predikere fremtidige øyebevegelser basert på tidligere bevegelse. Dette resulterte i en implementasjon av en Forecast N-BEATS modell, som ga svært gode resultater for fremtidig blikkprediksjon under sakkader.

Denne rapporten vil kapittelvis gå igjennom prosjektet, og gi leseren et innblikk i prosessen bak prosjektet og resultatene oppnådd. Først vil all relevant teori bli dekket for å gi leseren den nødvendige teoriske kunnskapen. Gruppens valg av teknologi samt metode blir videre dekket, før en gjennomgang av resultatene blir presentert for leseren. Avslutningsvis vil rapporten diskutere rundt observasjoner gjort under prosjektets gjennomførelse, samt resultatene i seg selv. Dette skal lede til et godt grunnlag for videre forskning på eyetracking.

Abstract

Eye tracking is the process of understanding where an individual is looking, what they're looking at and for how long they look at a certain spot. Our eyes are one of our primary tools we use in decision making. Researchers use eye tracking to study human behavior as it's the only way to observe, measure and understand visual attention objectively and accurately. This bachelor thesis and accompanying report builds upon a previous bachelor thesis from 2020, by Moan, Nygård og Ramsberg. The previous project focused on setting up scenes in VR for collecting eye tracking data, and then implementing several algorithms for the best possible classification of the data. The current project picks up in many ways where that project left off, but at the same time takes the research in new directions. The process has led to a scalable research project, which is designed for further development. The end results presented in this report suggest that it is possible to classify both visual input and users of VR headsets, with machine learning trained on the associated eye data. Furthermore, data can also be used to predict future saccadian movements.

One of the major constraints on the project in 2020 came from the collection of sufficiently varied data during the corona pandemic. This was however not a limiting factor during the implementation of the project which allowed the collection of a solid data base of eye tracking data. All the new data was used for the analysis of the classification algorithms, and it was shown that there are large differences in how they differentiate between saccades and fixations. This applies to both among the algorithms themselves, as well as in comparison of different scenes. This was the background for the development of four new experiments, which are designed to provoke different types of eye movements that are represented in the literature.

More than 300mb of eye data has been collected for the analysis. Bu using machine learning and the data from the new scenes, the group developed models that would be able to predict which scene the data belonged to. The CNN model provided a good starting point, with a precision over 60 percent for classifying scenes. The best results came from the RNN model, which on independent test sets classified scenes and test subjects with precision up to 90 percent. The RNN models were implemented in Unity for real-time classification, here the results were also good, with somewhat lower precision due to specific implementation details. It was also of interest to be able to predict future eye movements based on past movement. This resulted in an implementation of a Forecast N-BEATS model, which gave very good results for future eye prediction during saccades.

This report will go through the project in chapters and give the reader a good insight into the process behind the project as well as the results achieved. First, all relevant theory will be covered to provide the reader with the necessary theoretical knowledge. The group's choice of technology and method is further covered, before a comprehensive results chapter. Finally, the report will discuss observations made during the project's implementation, as well as among the results. this should all culminate to a solid basis for further research on eye tracking.

Innhold

Figurer

Tabeller

1 Introduksjon og relevans

Prosjektet blir gjennomført på vegne av 3D Motion Technologies, representert av oppdragsgiver Alexander Holt. Det ble beskrevet et ønske om å videreutvikle et system for å detektere og evaluere oppmerksomhet i virtuell virkelighet, for bruk i videre forskning. Som følge av den raske utviklingen innenfor VR-teknologi, har den et stort potensial for å bli et nyttig og økonomisk overkommelig verktøy. Forskningsprosjektet i denne oppgaven har gått over flere år, og har samtidig flere parallelle prosjekt. En målsetnign for dette arbeidet er å utforske muligheten til å kunne benytte eye-tracking for å oppdage sykdommer i tidligere stadier av sykdomsforløpet. Å utvikle og tilgjengeligjøre denne typen teknologi er i tråd med verdiene i 3D Motion Technologies sitt arbeid for velferdsteknologi.

Dette prosjektet bygger videre på arbeidet gjort av Moan, Nygaard og Ramberg sin bacheloroppgave fra 2019. I deres arbeid utforsket de innsamling av øyedata i Unity, og benyttet algoritmer for å klassifisere og evaluere den innsamlede øyedataen. Formålet med arbeidet som presenteres i denne rapporten, er å først gjøre en evaluering det tidligere arbeidet. Det er ønskelig å videreutvikle eksperimentene og bygge et større og bedre datagrunnlag, som kan brukes i både evaluering og videre forskning. Å klassifisere øyedata har vist seg å være en vanskelig disiplin, men med et større datagrunnlag kan det være mulig å benytte seg av maskinlæring på spesielt to områder.

- Klassifisere øyedata basert på hva brukeren ser på, og hvem som er brukeren
- Predikere adferden til øyedataen basert på dens tidligere bevegelsesmønster

Overordnet kan forskningsspørsmålet til dette prosjektet beskrives som:

Hvilke betraktninger må en gjøre for å utvikle gode eksperimenter og innsamlingsmetoder for øyedata i VR, og hvordan kan man håndtere og evaluere denne dataen ved hjelp av maskinlæring.

1.1 Dokumentstruktur

2 **Teori og relevant litteratur**: Oversikt over relevant litteratur og tidligere arbeid innenfor øyedata og maskinlæring. Begreper knyttet opp mot blikkdata og øyets bevegelse blir beskrevet, sammen med en teoretisk gjennomgang av evalueringsmetodene.

3 **Metode og teknologi**: I denne delen kommer først en redegjørelse av valg av teknologier og metodikk. Videre blir prosessen beskrevet, fra datainnsamling og behandling, til evaluering og implementasjon.

4 **Resultater**: Her presenteres gruppens resultater for arbeidet. Det blir også presentert resultater fra bruk av modellene og hvorvidt de egner seg. Gruppens ingeniørfaglige resultater og effektmål vil i stor grad være knyttet mot disse vitenskapelige resultatene. Til slutt blir gruppens administrative resultater kort presentert.

5 **Diskusjon**: Gruppens gjennomførelse og endelige resultater fra arbeidet blir omfattende diskutert i dette kapitelet. Samtidig blir dette satt inn i et større ingeniørfaglig perspektiv, slik at videre arbeid og hensyn også kan presenteres.

6 **Konklusjon**: Avslutningsvis gjør gruppen rede for om arbeidet i prosjektet ga svar på forskningsspørsmålet.

1.2 Akronymer og forkortelser

- VR : Virtual Reality
- CNN : Konvolusjonalt nevralt nettverk
- ONNX: Open Neural Network Exchange

- HMD: Head-Mounted Display
- NLP: Natural Language Processing
- LSTM: Long Short-Term Memory
- CNN: Convolutional Neural Network
- RNN: Recurrent Neural Network
- VOR: Vestibulo-Ocular Reflex
- I-VT: Identification by Velocity-Threshold
- I-DT: Identification by Dispersion-Threshold
- I-MST: Identification by Minimum Spanning Tree
- FIR: Finite Impulse Response
- NH: Nyström-Holmqvist
- I-HMM: Identification by Hidden Markov Model
- N-BEATS: Neural Network Architecture for Time-Series Rorecasting
- ANN: Artificial Neural Network
- KPI: Key Performance Indicators
- MSE: Mean Squared Error
- RSME: Root Mean Squared Error
- MAPE: Mean Absolute Percentage Error
- API: Application Programming Interface
- IPD: Inter Pupillary Distance
- GPU: Graphics Processing Unit
- ARIMA: Autoregressive Integrated Moving Average

1.3 Definisjoner

- Sanntid: Tilnærmet øyeblikkelig respons [21]
- Deteksjon: Oppdagelse av input for klassifisering [26]
- Prediksjon: Output fra maskinlæringsmodell som er trent på historisk data, som prøver å si noe om fremtidig data [26]
- Tidsserie: Datasett med en tidsmessig dimensjon[27]
- Seed: En initialiseringstilstand til en pseudo-tilfeldig tallgenerator[31]

2 Teori og relevant litteratur

Arbeidet som er gjort i dette prosjektet er en videreføring av Moan, Nygård og Ramberg sitt prosjekt 'Eyetracking i VR' [36]. I deres oppgave ble det gjort en omfattende litteraturstudie [37]. Mye av det teoretiske grunnlaget for denne oppgaven står beskrevet i deres dokument. Som følge av forskjellige problemstillinger og ny kunnskap på området, er det både nødvendig å presentere nye konsepter, samt spesifisere tidligere benyttet teori for dette prosjektets formål. I tillegg er det teoretiske konsepter som er så sentrale for arbeidet, at de av nødvendighet må beskrives i begge rapportene.

2.1 Tidligere relevant arbeid

Gruppens arbeid og prestasjoner kommer som konsekvens av tidligere arbeid med øyedata samt maskinlæring. Disse har banet vei for videre utvikling og forskning på området, og har vært med på å danne det teoretiske grunnlaget for også dette arbeidet.

2.1.1 Virtuell virkelighet (VR)

Allerede i 1998 beskrev J.M Zheng og K.W Chan virtuell virkelighet som et brukergrensesnitt som simulerer et realistisk virtuelt miljø som brukeren kan bevege seg i og interagere med [62]. Hensikten er å skape en illusjon av tilstedeværelse. Dette kan blant annet gjøres ved hjelp av VR-hodesett (HMD) som stenger ut de faktiske omgivelsene, og skal gi en realistisk og overbevisende representasjon av det virtuelle miljøet. Utvikling på området har gjort teknologien mer realistisk, og mer tilgjengelig for privatpersoner.

2.1.2 Blikkdeteksjon i VR

Som presentert av Moan, Nygard og Ramberg er ikke blikkdeteksjon i seg selv et nytt domene [37]. Observasjoner av menneskeøyet har blitt brukt i studier siden 1800-tallet. Blikkdeteksjon i forskning går ut på å benytte observasjonene til å si noe hvor blikket til en bruker befinner seg til en gitt tid, og er nyttig teknologi i alt fra psykologi til markedsføring [34]. Tradisjonelle studier av denne arten har benyttet skjermer og kamera for å fange blikket til brukeren [47]. Dette kommer frem i litteraturen, for eksempel fra Dalveren og Cagiltay sin studie om klassifisering av øyebevegelser [35]. Samtidig har samtaler med relevante personer som Ali Alsam gitt innsikt i denne typen arbeid [64].

Blikkdeteksjon i VR er en disiplin som byr på egne fordeler og utfordringer. Det benyttes to kamera som er rettet mot hvert sitt øye. Disse måler forskjellige parametere som blir presentert senere i rapporten, men mest sentralt måles retningen til brukerens pupiller relativ til sentrum av hodesettet. Clay, P König og S König beskrev i sin rapport fra 2019 utfordringer og potensiale med å benytte virtuell virkelighet i studier om blikkdeteksjon [14]. Det største potensiale for forskere ligger i muligheten til å i stor grad kontrollere testmiljøet, samtidig som at miljøet oppleves som realistisk av brukerne som benytter systemet.

2.1.3 Evaluering av øyebevegelser med algoritmer

Introduksjon, implementasjon og teorien bak relevante algoritmer for klassifikasjon av øyebevegelser ble dokumentert i hovedrapporten til Moan, Nygaard og Ramberg. Deres implementasjon ble benyttet som en del av dette prosjektet, og teorien beskrevet ble benyttet for evaluering av deres arbeid.

Forskjellige metoder for å evaluere brukbarheten til algoritmer som klassifiserer øyebevegelser eksisterer. I 2019 viste Dalveren og Cagiltay blant annet at *Friedmantesten* kan benyttes for å se på

ulikheter i klassifikasjoner på tvers av algoritmer, samtidig som den presenterer ulikheter i klassifikasjonsmetodene [35]. Dalveren og Caglitay argumenter samtidig at bruk av algoritmer nå er akseptert som den eneste praktiske metode for å klassifisere øyebevegelser på større mengder data. Dette begrunnes med at prosessen med manuell klassifisering er svært tidkrevende. Dette er også en av konklusjonene til Moan, Nygaard og Ramberg.

2.1.4 Klassifisering av øyedata med konvolusjonelle nevrale nettverk

Konvolusjonelle nevrale nettverk er et anvendbart og fleksibelt verktøy i klassifikasjonsproblemet. I 2018 benyttet Y. Yin, C. Chuan, J. Chakraborty og M.P McGuire et konvolusjonelt nevralt nettverk til å klassifisere øyedata med tanke på to forskjellige aspekter; klassifisering av forskjellige grafiske brukergrensesnitt, deretter klassifisering av nasjonaliteter til brukerne [61]. Artikkelen viser et eksempel på hvordan feature engineering kan benyttes ved å visualisere den innsamlede øyedataen, for bruk i det konvolusjonelle nevrale nettverket. Denne teknikken skal hente ut de mest relevante dataene for bruk i treningen. Resultatet av dette arbeidet var en presisjon på over 80 prosent ved klassifisering av to forskjellige typer grensesnitt. De klarte også å skille mellom Amerikanske og Saudi Arabiske nasjonaliteter med en presisjon på over 70 prosent, i hovedsak basert på brukerens lesevaner. I konklusjonen ble det beskrevet et behov for å benytte den temporale parameteren i feature engineeringen, og det argumenteres for at dette vil gjøre resultatene til en slik type modell bedre.

2.1.5 Reccurent Nueral Nettverk i Natural Learning Processing

Long Short-Term Memory ble i 1997 utgitt av S. Hochreiter og J. Schmidhuber [22], og løste for første gang behovet for langtidsminne i Reccurent Nueral Network. I deres artikkel beskrev de blant annet dens potensiale for bruken i NLP-problemer.

Nytten av LSTM har i senere tid blitt vist i flere publikasjoner. Blant annet viste Graves og Schmidhuber at lagring av informasjon over tid gjorde LSTM til et egnet verktøy for identifikasjon av sekvensiell data, slik som først antydet i Hochreiter og Schmidhuber sin originale artikkel [20]. Dette ble utgangspunktet for Graves i hans bok fra 2012, 'Supervised Sequence Labeling with Recurrent Nueral Networks' [19]. Blant annet viser Graves hvordan Sequence Labeling i NLP kan gjøres med *RNN*, sammen med den spesifikke anvendelsen av LSTM. Her vises også LSTM sin håndtering av vanishing gradient problemet, som gir svært gode resultater i presisjon og hastighet.

2.1.6 Reccurent Nueral Nettverk i øyedata

I 2019 foreslo S. Sims, V. Putnam og C. Conati RNN som et alternativ med begrenset behov for feature engineering i klassifikasjonsproblemet med øyebevegelsesdata. Sims, Putnam og Conati argumenterte for at klassifisering med RNN var bedre enn tidligere benyttede metoder for trening og evaluering av rådata fra øyet [52]. Denne rådataen inkluderte blikkets retning i x- og y-posisjon, øyets posisjon i kameraet, pupillstørrelse, og øyets distanse fra kameraet. Forsøket gikk ut på å detektere forvirring hos testpersoner basert på den innsamlede dataen. Fra sin forskning konkluderte de med at RNN kunne benyttes effektivt på rådata fra øyet uten behov for feature engineering og diverse bildeaugmentering, slik som med konvolusjonelle nevrale nettverk. Samtidig beskrev de et behov for å utforske forskjellige hyperparameteres påvirkning i større grad enn de gjorde selv, slik at en potensielt kunne få bedre resultater med liknende modeller.

2.1.7 Støyreduksjon

Målingsdata ved bruk av eyetracking, vil alltid inneholde støy [41]. Dette kan komme fra ulike kilder, både fra måleinstrumentene selv og fra omgivelsene de benyttes i. Tobii, produsenten bak eyetracking systemet i HMD-et, argumenterer for at støyreduksjon er svært viktig for at mange av de ulike algoritmene skal kunne fungere.

Det eksisterer flere ulike metoder å redusere støy. Moving Average er en algoritme som benytter et glidende vindu. Hvert punkt i den originale dataen beregner et gjennomsnitt sammen med et gitt antall punkter før og etter. Gjennomsnittet blir den nye verdien etter behandling. En tilsvarende metode finnes også hvor median verdi benyttes, kalt Moving Median. En fordel med sistnevnte er at den ofte kan redusere mye lav amplitude støy uten å opprette 'falske' punkter, ettersom median verdi er faktiske verdier. En annen mulig fordel er at den ikke utjevner endringen i verdier i like stor grad [41]. For øyebevegelse kan dette påvirke vinkelfarten, som står sentralt i flere klassifiseringsalgoritmer. Dersom samplingsraten er for lav og det eksisterer svært korte utslag som ikke er støy, kan Moving Median derimot by på problemer. Den legger bort alle verdier over og under den median verdi, og dette kan føre til at korte sakkader blir ignorert. Moving Average blir påvirket av alle punkter og er ikke utsatt for dette.

2.1.8 Gruppens tidligere arbeid innenfor maskinlæring

Gruppens tre medlemmer har tidligere arbeidet med maskinlæring gjennom spesielt to prosjekter. 'Anvendelse av Maskinlæringsmodeller for Identifikasjon av Språk', og 'Klassifisering av trafikkskilt i sanntid med OpenCV i Android'. Disse er ikke fagfelles vurdert, og er sådan ikke tilstrekkelige kilder for å kunne gi et teoretisk grunnlag for aktuell anvendelse av maskinlæring. Det er derimot arbeid som tillater gruppemedlemmene å gjøre erfaringsbaserte vurderinger på enkelte aspekter av også dette prosjektet.

I Anvendelse av Maskinlæring for Identifikasjon av språk, ble det demonstrert hvordan en PyTorch LSTM kunne gi gode resultater på klassifisering av språklig data, med noe høyere presisjon enn Gated Reccurent Units. Gruppen oppdaget at presisjonen økte lineært med hidden size, og at en læringsrate som tilpasset seg var gode for store datasett. Klassifisering av trafikkskilt i sanntid med OpenCV i Android viste at det var mulig å få god presisjon på en CNN modell, og samtidig benytte denne i annen teknologi for sanntidsklassifikasjon.

Prosjektoppgaver i løpet av et studieløp skal gi studenter erfaringer som de kan benytte i videre arbeid. Begge disse oppgavene er svært forskjellig fra prosjektarbeidet i denne oppgaven, og ingen resultater fra disse prosjektene er gjenbrukt. Det er for ryddighetens skyld likevel verdt å nevne disse arbeidene, og hvilke erfaringer som eventuelt har ledet til spesifikke valg av metode og implementasjon i dette prosjektet.

2.2 Mål for blikkdata

2.2.1 Romlig- nøyaktighet og presisjon

Romlig nøyaktighet er systemets evne til å korrekt måle en brukers blikk. Romlig presisjon beskrives som systems evne til å gjenskape en måling av blikkets posisjon [36] [23]. I et system med perfekt romlig presisjon, gitt at en brukers blikk ikke endrer seg over tid, vil systemet fortsette å gi samme avlesning av blikkdata. Etter hvert som en sesjon utføres er det forvente at den romlige presisjonen forverres ettersom bruk kan bevege headsettet på hodet til brukeren, og dermed gi forskjellige avlesninger. Det er derfor viktig å kalibrere HMD-et med jevne mellomrom.

2.2.2 Temporal Nøyaktighet

Temporal nøyaktighet er et mål på hvor regelmessig systemet får målinger som benyttes i blikkdeteksjon [33]. Det er i litteraturen blitt definert som variansen på målinger fra øyets bevegelse, til det registreringen av bevegelsen i systemet [23]. I praksis så betyr dette at desto jevnere og forutsigbart intervallene for datainnlesning skjer, desto høyere temporal nøyaktighet har systemet [36].

2.3 Øyebevegelser

Det eksisterer forskjellige øyebevegelser, og overordnet kan vi dele inn i to kategorier basert på formålet med bevegelser. Disse er bevegelser for å bevare blikket (eng.: maintain gaze) og endre blikket (eng.: change gaze) [40]. Videre deler man inn i underkategorier. For øyebevegelser med formål om å holde blikket er fikseringinger helt sentralt. Vestikubolære reflekser faller også under denne kategorien. På den andre siden har vi ulike typer bevegelser med formål om å endre blikket. Her har vi sakkader, jevn forfølgning (eng.: smooth pursuit) og vergens (eng.: vergence). I en oppgave som dette, hvor klassifiseringen er høyst relevant, er det essensielt med et godt teoretisk grunnlag for å kunne vurdere og skille klassifiseringer av disse kategoriene med bevegelser. Den neste delen av dette kapitelet vil derfor oppsummere og bygge videre på det teoretiske grunnlaget fra Moan, Nygaard og Ramberg sin studie.

2.3.1 Bevare blikk

2.3.1.1 Fiksering

Fiksering i seg selv er ikke direkte en bevegelse, men en handling øyet utfører når man holder øyene festet på et punkt over en gitt tidsperiode. Begrepet fiksering brukes for å beskrive perioden med relativ stabilitet der visuell informasjon behandles. Fikseringer varer typisk 200-300ms, men kan vare både kortere eller lengere. Varigheten til en gjennomsnittlig fiksering vil avhenge til en viss grad av konteksten fikseringen blir utført i. Som forklart av Moan, Nygard og Ramberg kan fiksering grupperes inn i ulike underkategorier avhengig av varigheten på fikseringen [36]. De beskriver tre undergrupper; korrigerende, omgivende– og fokale fikseringer.

Den korteste fikseringen er korrigerende fiksering, som forekommer mellom store og mindre sakkader. Omgivende fikseringer utspiller seg under en romlig orientering. En omgivende fiksering utføres for å orientere seg om lokasjonen til objekter i terrenget man befinner seg i. Omgivende fiksering er derimot for kort til å eksakt identifisere hva objektet er. Fokal fiksering har en lenger varighet en omgivende fiksering, og utspiller seg når en skal identifisere objektene som ble detektert under omgivende fiksering.

Ved en fiksering kan det virke som om øyet forholder seg helt i ro, men i virkeligheten er øyet aldri helt stille. Innenfor en fiksering kan det være både drift (langsomme endringer i plasseringen av fokus) og mikrosakkader. Som navnet antyder, er mikrosakkader akkurat som vanlige *sakkader*, bare langt mindre. Eksakt hvor liten en sakkade må være for å bli kategorisert som en mikrosakkade er en debatt uten noen definitivt entydig svar, som mye annet når det kommer til definisjoner av øyebevegelser. Men mindre enn 1 grad av synsvinkel er en rimelig vanlig grense å forholde seg til. De er av interesse for forskere av flere grunner. For å kunne oppdage mikrosakkader trenger man en eyetracker av høy kvalitet som trenger en rask samplingshastighet og svært høye presisjonsnivåer [17].

2.3.1.2 Vestibulo-okulær refleks (VOR)

Den vestibulo-okulære refleks (VOR)-mekanismen utfører bevegelser i øyet som et resultat av hodebevegelser. Disse øyebevegelsene sørger for å holde blikket stasjonært i forhold til omverden. En rotasjon av hodet (og med det en rotasjon av balanseorganet i det indre øret) vil tilføre en motsatt rotasjon i øynene. Dette kan eksempelvis være når en leser på en gjenstand som står i ro mens man beveger på hodet. Øynene til leseren vil stabilisere seg selv relativt til objektet, men beveger seg relativt til hodet. VOR-kansellasjon forkommer når man ønsker å skifte retningen på blikket sammen med hodebevegelsen [30].

2.3.2 Endring av blikk

2.3.2.1 Sakkade

En sakkade er en rask, konjugert øyebevegelse som flytter sentrum av blikket fra en del av synsfeltet til en annen. En konjugert bevegelse er en bevegelse hvor begge øynene beveger seg i samme retning. Sakkader utføres hovedsakelig for å orientere blikket mot et objekt av interesse. Disse øyebevegelsene kan forekomme både horisontalt, vertikalt eller skrått, men foretas oftest i en horisontal bevegelse og alltid på en lineær måte. Utførelsen av en sakkade kan både være frivillig etter eget ønske (f.eks. Ved skumlesing av en tekst) eller som en ufrivillig og refleksiv bevegelse (f.eks. Når en stimulus dukker opp utenfor periferien av ens synsfeltet, og resulterer i en orienteringsrefleks hvor man refleksivt beveger øynene). Under en sakkade vil man oppleve en sakkadisk synshemning som resulterer en svært redusert tilstand til å ta inn visuell informasjon under utførelsen av sakkaden [36]. Sakkader er hurtige bevegelser av øynene som muliggjør raske øyebevegelser mot visuelle, auditive eller taktile stimuli, og kan bevege seg opptil 700°/s. Sakkader benyttes til identifisering av punkter i omgivelsene våre for orientering eller utførelse av en ønsket oppgave. Initiering av en sakkade kan ta opp mot 200 millisekunder, og bevegelsen kan vare fra 30 opp til 120 millisekunder [6]. Sakkader sies å være ballistiske ettersom bevegelsesløpet er forhåndsbestemt fra sakkadens initiering, og en sakkade kan ikke avbrytes når bevegelsen først er utløst. Øynene klarer heller ikke reagere på påfølgende endringer i posisjonen til målet som skjer under sakkaden etter at en sakkade har blitt påbegynt. Dette gjelder både for å observere hvor et mål potensielt kan ha videre flyttet seg til, men også for å endre den prosjekterte banen for sakkaden.

2.3.2.2 Jevn forfølging

Utenom sakkader finnes det en annen måte man kan skifte plassering til blikket sitt, kalt jevn forfølging (eng: smooth pursuit). Jevn forfølging er en type øyebevegelse der øynene forblir fiksert på et objekt i bevegelse, som man følger med øyene etter hvert som objektet beveger seg. De fleste er ikke i stand til å starte en jevn forfølgelse uten et objekt i bevegelse en kan feste øyene på. Dersom prøver å fremtvinge en jevn forflytning uten bevegende objekt kan det resultere i en simulert bevegelse som kan virke som en jevn forflytning, men som i virkeligheten er en lang rekke med mindre sakkader.

Det er likevel fortsatt mulig å utføre forfølgelse uten et synlig mål, under spesifikke forhold. Om man vet hvilken retning et objekt vil bevege seg kan du starte forfølgelsen før bevegelsen faktisk starter, særlig dersom du vet spesifikt når bevegelsen vil starte. Det er også mulig å opprettholde forfølgelsen hvis et mål forsvinner et øyeblikk, spesielt hvis objektet forsvinner bak et annet objekt for en gitt tid. Også i totalt mørke kan man fortsatt utføre jevne forfølgelse ved hjelp av et proprioseptivt bevegelsessignal (f.eks. En finger) [12].

Hvis objektet som blikket følger beveger seg med vinkelhastigheter over 30°/s, har blikket en tendens til å utføre innhentende sakkader, ettersom objektet beveger seg for fort for jevn forfølging å holde følge. Jevn forfølgelse er asymmetrisk bevegelse, altså man har en tendens til å være bedre på horisontal enn vertikal jevn forfølgelse, spesielt definert av ens evne til å forfølge jevnt uten å måtte utføre innhentende sakkader. De fleste er også bedre til å følge objekter med blikket nedover fremfor oppover. I motsetning til sakkader, som er ballistiske bevegelser og ikke lar seg forandre under selve bevegelsen, vil jevn forfølging modifisere bevegelsen etter hvert som objektet flytter på seg. Skal noteres her at jevn forflytning har en kort ballistisk periode i det forflytningen blir initialisert og man først finner objektet man ønsker å forfølge.

2.3.2.3 Vergens

Vergens er en bevegelse i begge øynene i motsatt retning av hverandre. Den beskriver hvorvidt øynene konvergerer eller divergerer. Dersom et objekt beveges seg mot ansiktet ditt, vil øynene rotere innover mot hverandre for å kunne holde fokus på objektet, også kalt å konvergere. Når øynene ser på et objekt beveger seg vekk, vil øynene rotere utover mot ørene. Dette blir kalt å

divergere. Konvergens og divergens er unike øyebevegelser da dette er de eneste øyebevegelsene som ikke er konjugerte (at øynene beveger seg i samme retning). Disse øyebevegelsene er diskonjugerte [6].

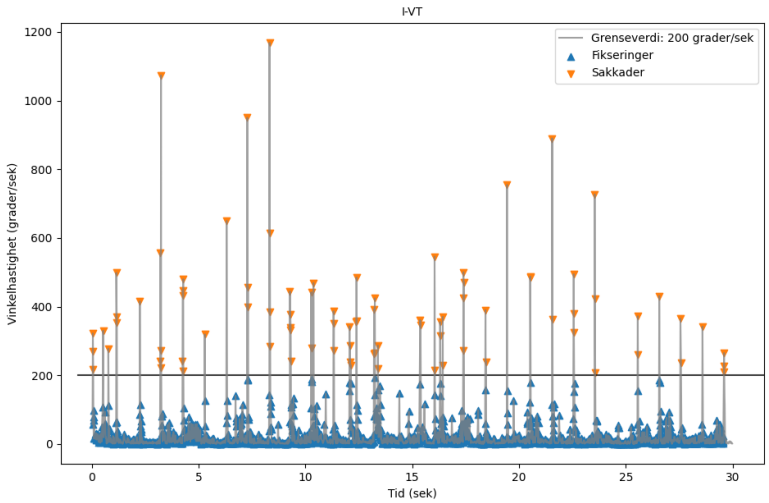
2.4 Evaluering av øyedata

2.4.1 Okulomotoriske algoritmer

I sin oppgave gjorde Moan, Nygård og Ramberg rede for de forskjellige tradisjonelle og moderne okulomotoriske algoritmene som ble benyttet i sin analyse [36]. I deres oppgave var disse algoritmene helt sentrale i oppgavegjennomførelsen. Selv om vi i denne oppgaven ikke i like stor grad fokuserer på disse algoritmene, er de likevel en viktig del av det teoretiske grunnlaget som diskusjonene i denne oppgaven tar utgangspunkt i. De blir også benyttet for klassifikasjon av data, som videre benyttes i analysen. Et generelt teoretisk grunnlag ansees derfor som hensiktsmessig, og en oppsummering av Moan, Nygård og Ramberg sin omfattende beskrivelse vil derfor presenteres:

I-VT Algoritmen (Identification by Velocity-Threshold) grupperer øyedataen i to basert på en terskelverdi for vinklehastigheten til øyet. Utvikleren definerer ønsket terskelverdi, hvor dataen med vinkelhastighet over denne verdien klassifiseres som sakkader, mens alt under klassifiseres som fikseringer [51].

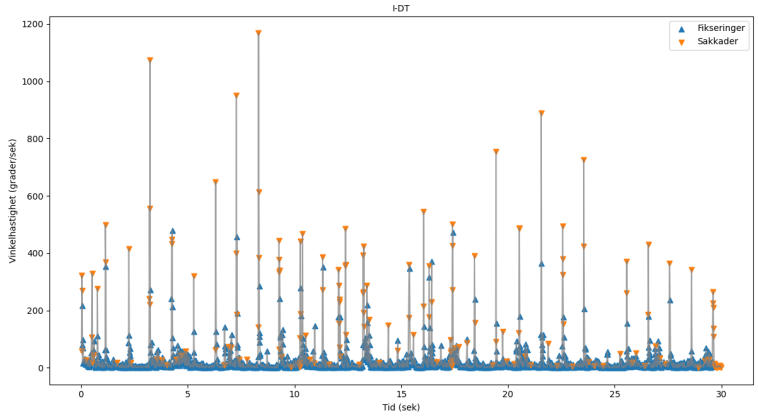
Figur 1: I-VT Klassifisering



IVT-Klassifisering på datasett: taxi-tour-still-001

I-DT Algoritmen (Identification by Dispersion-Threshold) finner fikseringer i øyedataen basert på den romlige spredningen. Dette gjør den ved å se om den romlige spredningen over tid er lavere eller høyere enn en gitt grenseverdi. I tillegg så settes en nedre grense for varighet av en fiksering, for å forhindre at urealistiske korte fikseringer klassifiseres [51].

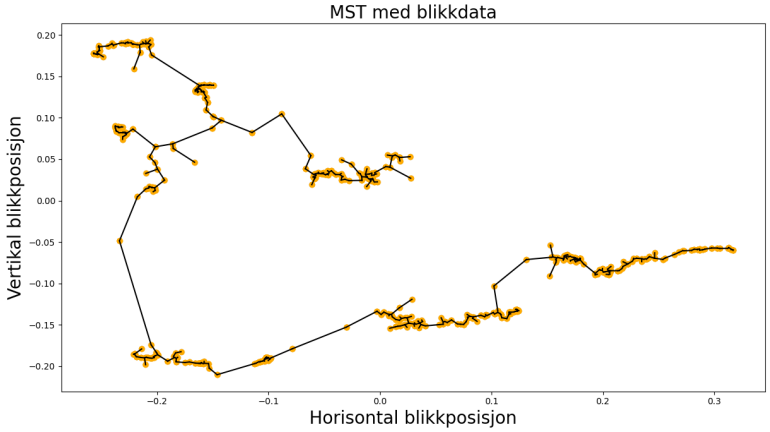
Figur 2: I-DT Klassifisering



IDT-Klassifisering på datasett: taxi-tour-still-001

I-MST Algoritmen (Identification by Minimum Spanning Tree) er en algoritme som baserer seg på romlig spredning. Den bygger et minimalt spennetre fra blikkoordinatene over tid. Algoritmen bruker blikkoordinater og retning til å lage en spenntre bygget opp av noder. Spenntreet er koblet sammen slik at summen av kantene mellom nodene skal være så lav som mulig. Figur 3 er spenntreet visualisert. På figuren kan man se kanter av ulik lengde mellom nodene, hvor fikseringer har kortere kanter enn sakkader [51].

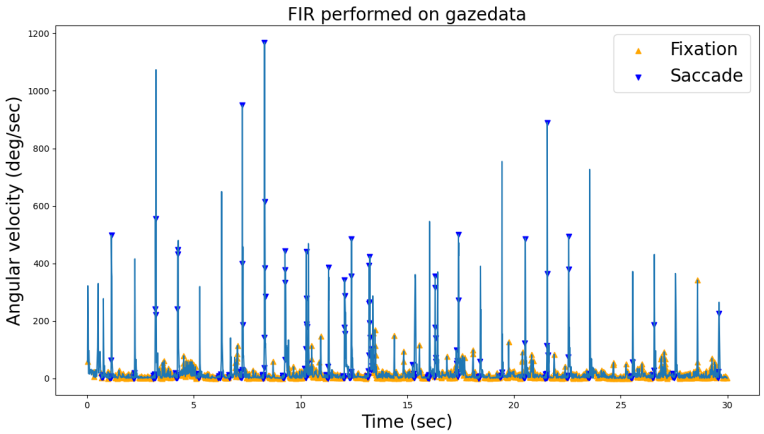
Figur 3: MST-Klassifisering



MST-Klassifisering på datasett: taxi-tour-still-001 (Visualisert med blikkets x- og y-posisjon)

FIR Algoritmen (Finite Impulse Response) er en algoritme som benytter FIR filtrering til klassifisering av øyebevegelser. Filtreringen kalles ofte for 5- eller 7-tapp. Navnet kommer av antallet numeriske elementer filteret er implementert til å håndtere. Algoritmen bruker kunnskap rundt oppførselen til sakkader med tanke på vinkelhastighet og amplitude. FIR algoritmen iterer over datasettet med en gitt grenseverdi for vinkelhastighet. Det er en stykkevisprosess hvor algoritmen tar punkter fra et gitt tidsintervall av gangen, og marker stykket enten som sakkade eller fiksering for å så ta et nytt stykke av datasettet og repetere prosessen [16].

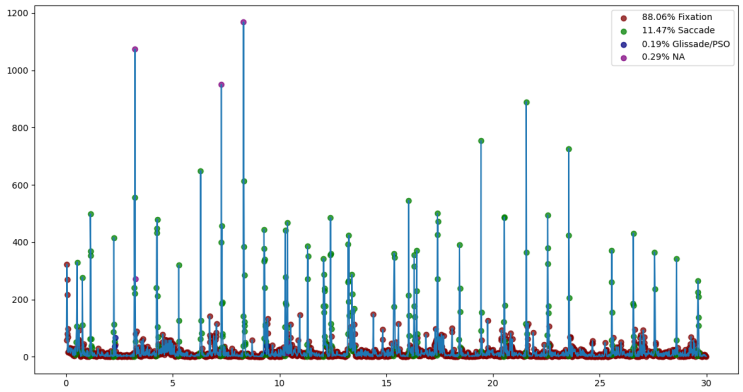
Figur 4: FIR-Klassifisering



FIR-Klassifisering på datasett: taxi-tour-still-001

NH Algoritmen (Nyström-Holmqvist) belager seg i likhet med I-VT og I-DT på å grenseverdier for klassifisering av blikkdata. Men ulik de tidligere algoritmene hvor bruker selv setter en grenseverdi vil NH algoritmen dynamisk regne ut egne grenseverdier for klassifisering av dataen. Algoritmen benytte seg også av Savitzky-Golay filtrering på hver romlig komponent for å dempe støy i datasettet. Algoritmen bruker vinkelhastighet og akselerasjon. Den dynamiske grenseverdien beregnes ut ifra total støy i datasettet i tillegg til å mindre lokale utvalg rundt punkter som blir ansett for å være en sakkade [39].

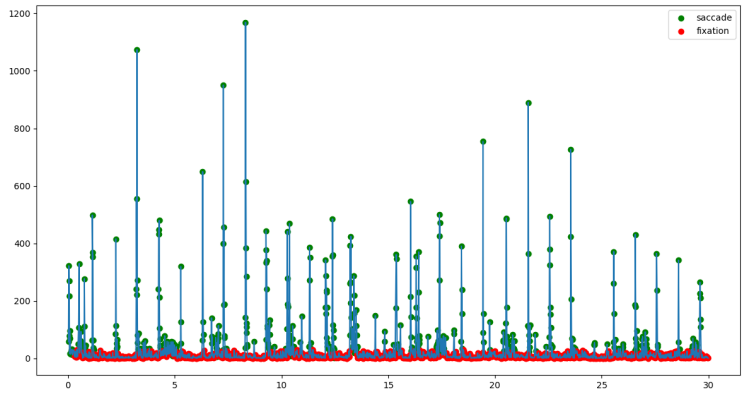
Figur 5: NH-Klassifisering



NH-Klassifisering på datasett: taxi-tour-still-001

I-HMM Algoritmen (Identification by Hidden Markov Model) Algoritmen bygger på en underliggende Markov modell. Denne modellen vil tildele en tilstand til en stokastisk verdi. I sammenheng med denne algoritmen vil disse tilstandene være fiksering og sakkader. Modellen ønsker å bestemme fremtidig tilstand basert på den nåværende tilstand, i noe kalt en Markov-rekke. I-HMM algoritmen trener en Markov-modell til å utføre tildeling av tilstand til datapunktene i et datasett [51].

Figur 6: IHMM-Klassifisering



IHMM-Klassifisering på datasett: taxi-tour-still-001

2.4.2 Friedmantesten

Friedmantesten er en metode for å få et statistisk mål på likhet gitt forskjellige tester, og egner seg når en ikke har en gitt fasit. Dette kan brukes til å måle eksempelvis likheter mellom pasienters reaksjon på forskjellige typer allergitester, samtidig som det gir et tall for usikkerheten knyttet til observasjonene i testen [25].

k: ser på antall kategorier som skal måles opp mot hverandre (for eksempel pasienter)

n: antallet tester som kjøres per kategori, desto flere tester desto mindre blir usikkerheten.

For hver test rangeres hver enkelt kategori etter forekomster på de forskjellige testene. Disse summeres og beskrives som T.

Friedmanstatistiskken F beskrives matematisk som

$$F = \frac{12}{nk!(k+1)}(T_1^2 + T_2^2 + \cdots + T_k^2) - 3n(k+1) \tag{1}$$

p-verdien beregnes: verdier lavere enn 0.001 for konfidens på 95 prosent.

2.5 Maskinlæring

Maskinlæring blir sett på som en del av fagfeltet kunstig intelligens. Det som kjennetener maskinlæring er at data benyttes for å lære opp en datamaskin slik at den senere kan gjøre egne prediksjoner uten å bli eksplisitt instruert av mennesker. For å trene opp en god maskinlæringsmodell, er viktig med et datasett som er stort og bredt nok til å kunne gi generelle prediksjoner. Valg av klassifiseringsalgoritme er ulik utifra hva formålet med modellen er. Ulike klassifiseringsproblem kan ha ulike behov for datatyper og datamende [32][48].

2.5.1 Kunstige Nevrale Nettverk

Begrepet Kunstige Nevrale Nettverk (ANN: Artificial Neural Network) er avledet fra biologiske nevrale nettverk som utvikler strukturen til en menneskelig hjerne. I likhet med den menneskelige hjernen som har nevroner koblet til hverandre, har kunstige nevrale nettverk også nevroner som er sammenkoblet med hverandre i forskjellige lag av nettverkene. Og akkurat som hvordan nevronene

i nervesystemet vårt er i stand til å lære av tidligere data, er ANN på samme måte i stand til å lære av dataene og gi svar i form av prediksjoner eller klassifiseringer.

ANN består av tre eller flere lag, som alle er sammenkoblet. Det første laget består av input nevroner. Disse nevronene sender data videre til de dypere lagene, som igjen vil sende de endelige output dataen til det siste ut-datalaget. Alle de indre lagene er skjult og dannes av enheter som adaptivt endrer informasjonen som mottas fra lag til lag gjennom en rekke transformasjoner. Hvert lag fungerer både som et input- og utgangslag som lar ANN forstå mer komplekse oppgaver. Samlet kalles alle disse indre lagene det nevrale laget.

ANN er ikke-lineære statistiske modeller med et komplekst forhold mellom inputs og outputs for å oppdage nye mønster. En rekke oppgaver som bildegjenkjenning, talegjenkjenning, maskinoversettelse samt medisinsk diagnose tar i bruk slike kunstige nevrale nettverk.[1]

2.5.2 Over- og undertilpasning

En modell er undertilpasset når en den gir dårlige resultater og presisjon på treningsdataen. Generelt kan man si at en maskinlæringsalgoritme sliter med undertilpasning når den ikke klarer å fange opp den underliggende trenden til datasettet. Modellen ikke klarer å fange opp forholdet mellom input eksemplene (data man ønsker å belage en prediksjon på, gjerne kalt X) og målverdiene (selve verdien man predikerer, ofte kalt Y). Undertilpassning betyr som regel at modellen eller algoritmen man har valgt ikke nødvendigvis passer godt nok til dataene. Dette skjer også når man har mindre enn ønskelig mengde med data til å trene en modell på.

Det motsatte av dette kalles overtilpassing. Når en modell overtilpasser treningsdataen vil man se at modellen predikerer bra på treningsdataene, mens resultatene ikke lar seg gjenskape på et evalueringsdatasett. Dette kommer når modellen begynner å lære av støyen og unøyaktigheter som befinner seg i treningsdatasettet. Denne tilførselen av detaljer og støy resulterer i at modellen ikke kategoriserer dataen riktig. Ved tilfeller av overtilpassning kan tiltak som feature selection, støy reduksjon av data eller øke mengden treningsdata være nyttig [9][3].

2.5.3 Tapsfunksjon og gradient descent (optimalisering)

Tapsfunksjon et mål på hvor god prediksjonsmodellen gjør det når det kommer til å kunne predikere forventet utfall/verdi. Vi konverterer læringsproblemet til et optimaliseringsproblem, definerer en tapsfunksjon og optimaliserer deretter algoritmen for å minimere tapsfunksjonen.

Hovedsakelig kan tapsfunksjoner klassifiseres i to hovedkategorier avhengig av hvilken type læringsoppgave man har å gjøre med. Disse funksjonsgruppene er regresjonstap og klassifiseringstap. I klassifisering prøver man å predikere en output verdi fra et datasett med endelige kategoriske verdier. Det vil si gitt et stort datasett med bilder av sifre, for å så kategorisere dem som ett av sifrene fra 0-9. Regresjon derimot omhandler å forutsi en kontinuerlig verdi for eksempel gitt gulvareal, antall rom, størrelse på rom, forutsi prisen på rom. Typsike tapsfunksjoner er Mean Squared error, Mean Absolute Error og Log-Likelihood Loss [2].

Optimalisering er prosessen med å justere hyperparametere for å minimere kostnadsfunksjonen ved å bruke en av en optimaliseringsmetode. Det er sentralt for å utvikle en godt ytende modell å minimere kostnadsfunksjonen, ettersom den beskriver avviket mellom den sanne verdien av den estimerte parameteren og det modellen har forutsagt. Mens tapsfunksjonen refererer til feilen for et enkelt treningseksempel, refererer kostnadsfunksjonen til et gjennomsnitt av tapsfunksjonen over et helt treningsdatasett [4].

Gradient descent er en svært populær optimaliseringsmetode som blir brukt en rekke algoritmer som regresjon, nevrale nettverk og lignende for å lære vekter/parametere.

2.5.4 Aktiveringsfunksjoner

En aktiveringsfunksjon er en funksjon som blir lagt til i et kunstig nevralt nettverk for å hjelpe nettverket med å lære komplekse mønstre i datasettet. Sammenlignet med en nevronbasert modell som er i en menneskehjerne, er det aktiveringsfunksjonen på slutten som bestemmer hva som skal sendes til neste nevron. Det er akkurat det en aktiveringsfunksjon gjør i en ANN også. Den tar inn utgangssignalet fra forrige celle og konverterer det til en form som kan tas som input til neste celle.

Det er aktiveringsfunksjonen som klassifiserer data som brukbart eller ikke. Aktiveringsfunksjonen holder altså er svært viktig jobb, ettersom det ikke er all data i et datasett som er sentralt eller viktig for et gitt nevralt nettverk.

2.5.5 Konvolusjonelt Nevrale Nettverk

En stor gruppe av maskinlæringsmodeller er innenfor kategorien nevrale nettverk. Denne kategorien kjennetegnes ved at den består av et input lag, ett eller flere skjulte lag (eng.: hidden layers), og deretter et utgangslag. Hver node kobles til en vektor med en aktiveringsfunksjon. Hvis en verdi innenfor noden er over en gitt terskelverdi blir noden aktivert, før den sendes videre til neste lag. Hvilke noder som blir aktivert bestemmer hva prediksjonen for utgangslaget er.

Innenfor kategorien nevrale nettverk eksisterer det ulike typer. En mye brukt metode innenfor bildebehandling er konvulosjonelle nevrale nettverk (CNN). Konvulosjonelle nevrale nettverk lærer seg mønstre i bildene/input, såkalte lærbare vekter og skjevheter (eng.: weight, bias). Disse benyttes for å til slutt predikere gitte utgangsverdier. CNN benyttes ofte til klassifisering, data-synsoppgaver (eng.: computer vision) og optimalisering [24].

Konvolusjonelle nettverk har god ytelse sammenlignet med flere andre nettverk når det kommer til vurdering av bilder, tale og lyd. Innad i nettverket finner man i all hovedsak tre ulike lag:

1. Konvolusjons lag (eng.: Convolutional layer)
2. Samlings lag (eng.: Pooling layer)
3. Fullt sammenhengende lag (eng.: Fully-connected layer)

Et konvulsjonslag kommer ofte tidlig i modellarkitekturen. Dette påfølges av flere tilsvarende lag eller samlingslag. Samlingslag benyttes for å redusere kompleksiteten til input, og dermed gjøre det enklere å lære tendenser i data. Til slutt samles data gjerne i et fullt sammenhengende lag i nettverket. Flere lag i modellen øker kompleksiteten og øker dermed potensialet for å identifisere større deler input bilder. Tidligere lag fokuserer på enkle egenskaper som omriss og farge. Senere lag gjenkjenner større egenskaper som former og interessante elementer, slik at tiltenkte objekter til slutt kan identifiseres [50].

2.5.6 Reccurent Nueral Networks

2.5.6.1 Vanishing Gradient

I tilbakevendende nevrale nettverk (RNN) oppstår ofte problemstillingen at den har utfordring med å huske informasjon for langt tilbake i input sekvensen. En kan gjerne si at RNN har kort-tidshukommelse. Årsaken til dette er det såkalte 'vanishing gradient problem' [11]. Hovedessensen i problemet er at gradienten i modellen over tid blir svært liten, etter at man back propargater mange ganger. Det er gradienten som benyttes for å oppdatere vekter i modellen, og dermed lære nettverket. Dette gjør at lag med lav gradientverdi slutter å tilføre læring. Dersom input er lang og en ønsker å dra nytte av kunnskap langt tilbake i sekvensen, kan dette dermed være vanskelig.

2.5.6.2 Hidden State

Skjult tilstand (eng.: Hidden state) beskriver hvordan vekter skal tolke variabler fra en tidligere tilstand. Alle verdier i input sekvensen som ikke er lik null vil oppdatere hidden state gjennom en sigmoid funksjon fra forrige tilstand sammen med nåværende. Formelen viser måten dette regnes ut i tradisjonelle RNN [8].

$$h_t = \begin{cases} 0, & t = 0 \\ \phi(h_{t-1}, x_t), & otherwise \end{cases}$$

$\mathrm{h}_t = g(Wx_t + Uh_{t-1})(2)$

2.5.7 Long Short-Term Memory

For å løse 'Vanishing Gradient Problem' kan man benytte en utvidelse av RNN kalt LSTM (Long Short-Term Memory) [11]. LSTM benytter et aktiversingsfunksjonslag, også kalt porter. Portene består ofte av sigmoidfunksjoner eller tanh. De benyttes for å kunne bedre flyten av informasjon gjennom modellen. Portene lærer informasjon på tvers av sekvensene og kan da finne ut hvilke sekvenser som er nyttig. Viktig informasjon blir tatt vare på og lagret i en såkalt cell state.

Celle state er en type langtidshukommelse, og dette skiller den fra ordinære tilbakevendende nevrale nettverk. Den holder på informasjon som blir ansett for å være nyttig slik at den kan brukes i senere lag. Celle state kan på lik linje hidden state oppdateres før den sendes videre til neste celle. LSTM sender dermed videre både cell state hidden state til påfølgende lag [42][44].

Porten i LSTM er som følger:

1. (f) Forget gate/remember vektor - Porten bestemmer hvilken input den anser som nyttig og ikke, og lagrer informasjon deretter. Sigmoid blir benyttet for å predikere dette. Denne gir et tall mellom 0 og 1 utifra hvorvidt informasjonen bør taes vare på.
2. (i) Input gate/ save vektor - Denne porten velger hva som blir lagt inn i celle state, og dermed lagret i langtidshukommelse. Også her benyttes sigmoid aktiveringsfunksjon for å vekte nytteverdi av den tillærte informasjonen.
3. (g) Input modulation gate - Mulige kandidater for cell state legges til her. Siste trinn for input er en Hadamard produkt operasjon som kombinerer de to input portene for å danne en ny celle state.
4. (o) Output gate- Utifra celle state vil neste output lag, altså neste hidden state, bli generert av output porten.

Matematikken bak LSTM kan vises slik [46]:

Hvor: i(t), f(t), g(t) og o(t) er enkeltvis i rekkefølge: input gate, forget gate, input modulation gate og output gate. h(t) = hidden state, c(t) = celle state, x(t) = input. σ = sigmoid og $\odot$ = Hadamard produkt. T viser til variablenes tilstand i gitt tid t. t1 vil da referere til en tidligere tilstand til de ulike variablene (Eksempel: input gate benytter den innsendte hidden state i sin beregning, altså h(t-1)) [29].

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-i} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{3}$$

2.5.8 Time Series Forecasting

Forecasting er en metode for å utføre informerte prediksjoner ved hjelp temporal data og trendene observert i den dataen som input for predikere fremtidige hendelser. Metoden vil ikke nødvendigvis gi en eksakt prediksjon som man er vant til ved klassifisering, men heller en numerisk tilnærming for hva modellen antar vil skje i fremtiden basert på de forgående trendene i datasettet. Forecasting er ikke en bestemt type modell som for eksempel RNN, men heller en generell metode som kan bli implementert på mange ulike måter. Avhengig av datasettet kan sannsynligheten for forecastingen kan variere voldsomt. Dette gjelder spesielt temporale datasett der man håndterer variabler som kan ha store svingninger i verdi. Av den grunn vil et med større datasett, generelt sett føre til mer nøyaktig forecasting, ettersom det gir mer rom for observasjon av større trender i datasettet. Datasettene brukt i forbindelse med forecasting kalles ofte for tidsserier (eng.: time series). Navnet kommer av at datasettet har en tidsmessig dimensjon som binder datapunktene i datasettet sammen. Det er viktig at datapunktene er likt fordelt over tid for å gi de tilhørende verdiene samme tidsmessig relasjon til hverandre. Forecasting er avhengig av et temporalt datasett for å kunne observere trender og forandringer i et datasett over tid.

Utrykk som 'forecaste' eller 'predikere' brukes ofte om hverandre. Generelt inngår forecasting i en undergruppe under paraplybegrepet prediksjon. Analysen en modell gjør av det temporale datasettet ligger som nevnt til grunne for å finne videre utfall og antagelser om fremtidige datapunkter i datasettet. Denne tidsdimensjonen er en sentral forskjell mellom klassifikasjon som prediksjon og forecasting. Dette kommer av at mens forecasting benytter seg av tidligere og nåværende data for å finne ut utfallene for fremtiden, vil en klassifikasjon ikke nødvendigvis ha samme restriksjon. Klassifikasjon er ikke begrenset av noen form for tid, enten fortid eller nåtid. Forecasting derimot er helt avhengig av den temporale dimensjonen. Men kort sagt kan man si at alle forecastinger er prediksjoner, men alle prediksjoner er ikke nødvendigvis forecasting [53].

2.5.8.1 Nøkkelindikatorer

Ettersom forecasting ikke klassifiserer data, men heller predikerer en numerisk tilnærming for prediksjon som tidligere nevnt kan man heller ikke bruke presisjon på samme måte, som en metode for å måle ytelsen til modellen. Det er mulig å sette egne grenser for hva som regnes som riktig predikering eller ikke. Det er derimot mer vanlig at forecasting belager seg nøkkelindikatorer (KPI: key performance indicators) for å måle ytelsen til en modell [59]. En nøkkelindikator er en målbar verdi som kan fortelle oss hvor god en forecasting er på å predikere i forhold til de sanne verdiene. Det finnes flere ulike nøkkelindikatorer som man kan bruke for å måle ytelse. Valg av metode vil avhenge av dataen man ønsker å analysere. Det er også enkelte nøkkelindiatorer som er enkelre å tolke enn andre.

En av disse nøkkelindikatorene er gjennomsnittlige kvadratfeilen (MSE: Mean Squared Error). MSE forteller oss hvor tilpasset linje eller punkt er et gitt datapunkt(er). Dette skjer ved å ta avstandene fra punktene til regresjonslinjen (disse avstandene kalles 'errorene') og kvadrere dem. Kvadreringen benyttes for å unngå at negative og positive feil utligner hverandre. MSE kalles den gjennomsnittlige kvadratfeilen ettersom man finner gjennomsnittet av et sett med feil. Desto mindre Mean Squared Error er, desto bedre tilpasset er dataen. En annen nyttig mengde man kan benytte seg av er Root Mean Squared Error (RMSE). RSME betegnes som kvadratroten av den gjennomsnittlige kvadratfeilen. Denne nøkkelindikatoren er en av de lettere vurdere ettersom benevningen er den samme som på vertikal akse [55].

Den gjennomsnittlige absolutte prosentvise feilen (MAPE) er gjennomsnittet av de absolutte prosentvise feilene i prediksjoner. Feil er definert som den sanne verdi minus prediksjonsverdien. Prosent feil summeres uten hensyn til fortegn for å beregne MAPE. Denne målingsmetoden er enkel å forstå ettersom det gir feilen i prosent. Ettersom absolutte prosentvise feil brukes, unngås problemet med positive og negative feil som kansellerer hverandre. Mindre MAPE prosentverdi tilsvare bedre prediksjonsresultat. På bakgrunn av dette er MAPE er populært målingsmiddel for å måle prediksjoner i forecasting[54].

3 Valg av teknologi og metode

3.1 Teknologier

3.1.1 VR-headset (HTC VIVE Pro Eye)

Under prosjektet ble HTC sitt VIVE Pro Eye headset brukt til innsamling av øyedata. Vive Pro Eye bruker nær-infrarødt lys (NIR 850nm) for å skanne brukerens øyebevegelser. Headsettet har også ni IR-lysdioder og ett IR-kamera per øye (totalt 18 lysdioder og to kameraer). Sensorarrayet lar headsettet spore øyebevegelser med en nøyaktighet på 0,5 til 1,1 grader over hele headsettets 110-graders synsfelt. I tillegg til blikkets startpunkt og retning (sporer nøyaktig hvor en bruker ser), kan Vive Pro Eye samle inn data om pupillens posisjon, pupillstørrelse og øyeåpenhet. Det infrarøde kamera tar bilder med en frekvens på 120Hz, men skjermen inni selve headsettet er begrenset på 90Hz [56].

Med headsettet følger med to tilhørende kontroller til headsettet, samt to sensorer. Disse sensorene som er montert på tripods, og er med på å spore brukeren i den virtuelle verden når de benytter headsettet og kontrollerne. Selve headsettet er koblet til en datamaskin med en 5 meter lang kabel, og gir brukeren mulighet for å bevege seg i det fysiske rommet. Maskinen som headsettet er koblet opp mot kjører Windows 10 operativsystem.

3.1.2 SteamVR

SteamVR er et VR-rammeverk med flere støttefunksjoner og plugins. Det er dette rammeverket som lar en opprette de virtuelle rommene, og ved hjelp av infrarøde sensorer plassert rundt brukeren kan SteamVR spore både selve headsettet samt kontrollerne.

SteamVR har en Unity plugin for et smidig grensesnitt mellom SteamVR og spillmotoren Unity. Med SteamVR kan utviklere rette seg mot ett API som alle de populære VR-headsettene kan koble til. Videre administrerer SteamVR Unity Plugin hovedsakelig tre punkter for utviklere: lasting av 3d-modeller for VR-kontrollere, håndtering av input fra disse kontrollerne og estimering av hvordan hånden til brukeren ser ut mens de bruker kontrollerne [36][58].

3.1.3 Unity

Prosjektet implementerer en rekke eksperimentelle VR scener og miljøer alle med ulikt formål avhengig av den ønskede øyebevegelse man ønsker fremtvungen. Alle disse scenene er implementert i Unity spillmotor. Unity er en populær spillmotor å bruke i utvikling av både VR spill og generelle miljøer. Utvikling av programvare i Unity skjer hovedsakelig i C# [36].

3.1.4 SRanipal

VIVE Pro Eye har et eget programmeringsgrensesnitt, eller API, kalt SRanipal som lar utviklere spore brukernes øyebevegelser, noe som gir utviklere mulighet til å benytte seg av dataen lest av fra øyenbevegelsene. SRanipal inkluderer funksjoner for å måle øyebevegelser og tidsrelaterte data. Dette er data som pupillstørrelse, hvor åpent et øye er, tidsstempel, og hvor øyet peker i den virtuelle verden. SRanipal har data for både venstre og høyre øye, samt en kombinasjonsvektor for begge øynene. Plugins for Unity er inkludert i API'et.

SRanipal tilbyr også kalibrering for blikkdetektoren. Denne kalibreringen burde kjøres ofte for å forebygge ujevn data. Under bruk vil headsettet avhengig av scenen gjerne bevege seg rundt på brukerens hode, så for å oppnå mest jevnt dataresultat burde kalibreringen gjøres jevnlig. Spesielt ved tilfellet av en ny bruker av headsettet må det kalibreres. Dette kommer av at man må finstille headsettet enkeltvis til hver enkelt bruker, spesielt med tanke på hodestørrelse samt mellomrommet

mellom brukerens pupiller. Denne avstanden kalles gjerne IPD (Inter Pupilary Distance), og kan variere fra person til person [36].

3.1.5 Maskinlæring

Maskinlæring er et bredt begrep, hvor det er essensielt å ta informerte valg basert på teori og erfaringer for å kunne oppnå ønsket resultat. Som konsekvens av dette er det også viktig å redegjøre for valg av teknologier som benyttes for den valgte løsninger. Valg av maskinlæringsteknologier er gjort på bakgrunn av hva vi anser som aktuelle benyttelse av resultatet. Det er derfor vært fokus på godt støttede teknologier, slik at resultatet skal være så skalerbart som mulig med tanke på videre utvikling.

3.1.5.1 PyTorch

PyTorch er et populært bibliotek for å skrive maskinlæringsmodeller i Python. Det er et fleksibelt bibliotek som fasiliteter for dyp-læring av modeller. PyTorch sin støtte opp mot andre teknologier er en av de største fordelene med biblioteket, spesielt for et prosjekt som dette hvor det er ønskelig å kunne benytte resultatet fra forskningen til formål som ikke nødvendigvis er definert på dette tidspunktet i prosjektet. Blant annet så er det ønskelig å kunne implementere maskinlæringsmodellen i Unity [43]. Det er også aktuelt å kunne benytte seg av modellen med OpenCV for sanntidsklassifisering av billedlig data [7]. Støtte for CUDA gjør at det er mulig å benytte seg av GPU'en under trening, gitt at maskinen som trener har et NVIDIA GPU [38].

3.1.5.2 OpenCV

OpenCV er et open source bibliotek for blant annet sanntidsdeteksjon av bilder. OpenCV har støtte for bruk i Unity [18]. Selv uten behov for direkte implementasjon i Unity, kan OpenCV være nyttig for å preprosessere bilder som kan lagres som pickle-binærfiler ved hjelp av Pandas dataframe.

3.1.5.3 ONNX

Open Neural Network Exchange er et åpent format for å utveksle algoritmer og maskinlæringsmodeller på tvers av plattformer og verktøy. Flere populære maskinlæringsrammeverk, som PyTorch, TensorFlow, Keras og Matlab har mulighet for å eksportere modeller til dette formatet. Vi valgte å benytte oss av ONNX på ettersom av det var kompatibelt med Unity.

3.1.5.4 Barracuda

Barracuda er et lettvekts rammeverk for kjøring av maskinlæringsmodeller i Unity. Det kan kjøre nevrale nettverk både på grafikkort og prosessor. Rammeverket er under utviklingsfasen og kommer stadig med oppdateringer med utvidet støtte. Barracuda støtter ikke alle modelltyper og parametere, men de nyeste versjonene ga tilstrekkelig støtte etter noe justering av modellene våre. Valg av teknologien ble gjort på bakgrunn av at det var enkelt å sette opp og at det unngikk tyngre rammeverk.

3.1.6 Darts

Darts er et Python bibliotek for enkel manipulering og prognose av tidsserier. Bibliotektet inneholder en rekke modeller, eksempelvis ARIMA (Autoregressive integrated moving average) og dype nevrale nettverk [15]. Alle modellene kan benyttes på samme måte. Dette gjøres enkelt gjennom bibliotekets universale fit() og predict() funksjoner. Biblioteket gjør det også mulig å kombinere

prediksjoner fra flere modeller eller å ta hensyn til eksterne data. Dart støtter både univariate og multivariate tidsserier og modeller. De maskinlæring-baserte modellene kan trenes på potensielt store datasett som inneholder flere tidsserier [49].

3.2 Metode

3.2.1 Prosess

Arbeidet i denne bacheloroppgaven blir å anse som et forskningsprosjekt. Det inneholder deler av utvikling, både i form av utvikling av scener og implementasjon av modeller i statistikk og maskinlæring. Hensikten med utvikling av verktøyene er for å støtte under forskningen gruppen foretar seg. Gruppens resultater skal først og fremst være forskning samt verktøy og data som skal fasilitere for videre forskning. Konsekvensen av denne tilnærmingen til prosjektoppgaven, er at det er vanskelig å fastsette datoer fra starten i planleggingsfasen, da progresjon er vanskelig å forutsi. I tillegg gjør oppgavens natur det vanskelig å avgjøre hva som skal være endelige mål for forskningen, da resultater og betraktninger i løpet av prosjektarbeidet vil være med å bestemme hvor gruppen skal rette sitt fokus. Dette er for øvrig noe oppdragsgiver både er klar over og oppfordrer til. Mål og delmål endres underveis, likevel er det viktig å ha en jevn dialog for å holde gruppens og oppdragsgivers interesser i fokus gjennom hele prosjektforløpet. Sammen med oppdragsgiver og veileder Alexander Holt ble det avgjort at et verktøy som Gantt-diagram ikke nødvendigvis ville være egent for å måle progresjon. Likevel var det enighet i at et slikt verktøy vil være gunstig i starten av prosjektarbeidet, slik at gruppen kunne starte planlegging av hva en kunne forvente å måtte gjøre i løpet av våren. Her satt gruppen foreløpige delmål, og et estimat på dato for når en ønsket at disse skulle være ferdige. Det ble definert fem deler av prosjektarbeidet, hvor alle hadde tre til fire delmål for å beskrive hva gruppen så for seg måtte gjøres. Dette utgjorde gruppens fremdriftsplan.

1. **Oppstart (10. Jan - 28. Jan)**

   I oppstarten er fokuset først og fremst å bli kjent med litteraturen, det tidligere arbeidet, og utstyret som gruppen skal benytte seg av i løpet av prosjektet. Her inngår også de formelle arbeidskravene og dokumentasjonen som må på plass i denne fasen.

   (a) Visjonsdokument
   (b) Forprosjektplan
   (c) Utforskning av tidligere arbeid
   (d) Bli kjent med utstyr og programvare

2. **Datainnspilling (31. Jan - 4. Mars)**

   Neste steg gruppen ser for seg, er datainnsamling basert på de tidligere utviklede scenene. Samtidig planlegges det å utvikle egne, nye scener med data som kan klassifiseres.

   (a) Utvikle nye scener for innsamling av data
   (b) Teste systemet på gruppemedlemmer
   (c) Samle inn data fra uavhengige testpersoner
   (d) Begynne å behandle og klassifisere data

3. **Databehandling og modellering (7. Mars - 1. April)**

   Denne fasen skal gå ut på å klassifisere og behandle dataen som er samlet inn, samtidig som man kan gjøre en vurdering av de algoritmene som er implementert. Denne fasen ser gruppen for seg at arbeidet med maskinlæringsmodeller skal begynne.

   (a) Klassifisere og behandle data ytterligere
   (b) Sammenligne algoritmer

(c) Utvikle maskinlæringsmodeller

4. **Visualisering og forskning (4. April - 29. April)**

   Her skal gruppen forbedre og videreutvikle arbeidet som er gjort så langt, slik at en kan begynne å analysere resultatene.

   (a) Videreutvikle modeller og algoritmer
   (b) Analysere data og resultater
   (c) Grafisk grensesnitt for analyse (om nødvendig)

5. **Ferdigstilling (2. Mai - 18. Mai)**

   Denne delen går i all hovedsak ut på å skrive ferdig hovedrapport og annen nødvendig dokumentasjon. Den siste finpussen på analysen legges også i denne fasen.

   (a) Skrive ferdig dokumentasjon
   (b) Siste forbedring og analyse
   (c) Begynne på presentasjon (frist 27. mai)

Selv om prosjektforløpet som forventet ikke utspilte seg nøyaktig som planlagt, var dette de fundamentale milepælene gruppen måtte jobbe seg gjennom for å ende opp med sluttproduktet som presenteres i denne rapporten. Den viktigste delen av prosessen vurderer gruppen til å bli valgene av metode som beskrives videre i dette kapitelet. Hvordan prosessen eventuelt ble annerledes enn først planlagt, blir beskrevet senere, i resultatdelen.

3.2.2 Datainnsamling

For datainnsamling av øyedata i VR-miljø benyttes SRanipal Eye Framewrok, sammen med Moan, Nygaard og Ramberg sin implementasjon og av LogDataReader beskrevet i deres rapport [36]. Ved hjelp av disse verktøyene er det mulig å samle og lagre en rekke øyedata. Det relative tidspunktet til eksperimentstart lagres også for vært punkt, slik at en kan regne ut nyttige parametere for bruk i okulomotoriske algoritmer, maskinlæring og generell kvantitativ analyse. Mer om hvordan databehandlingen foregikk kommer senere i punkt 3.2.3.

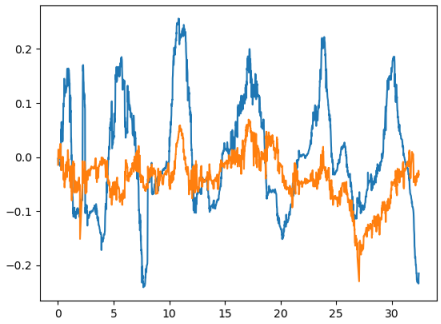
Dataen samles inn om lag 90 ganger i sekundet, og lagres i en egen csv-fil for hver kjøring. Denne navngis og plasseres automatisk etter parameterne utvikleren bestemmer i Unity editor.

3.2.2.1 Videreføring av tidligere arbeid

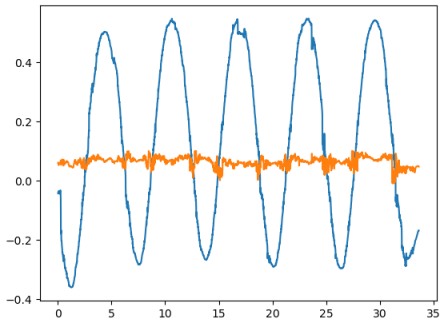
Fra tidligere var det samlet inn omkring 30MB data fra totalt fem scener. Dette var scenene 'Jevn forflytning', 'Sakkadedeteksjon', 'Huskestue', 'PIN-kode' og 'Drosjetur'. På dette tidspunktet i 2020 var koronarestriksjonene svært begrensende for prosjektarbeidet, og som konsekvens av pandemien ble både scenene og mengden tilhørende data beskrevet som mangelfulle [36]. Dette er en betraktelig mindre faktor i 2022, og lar spesielt to aspekter av videreføringen bli utforsket ytterligere. Den første går på systematisk datainnsamling av de eksisterende scenene med langt flere testpersoner for å verifisere resultater og utforske mulige mangler. Den andre handler om å bruke tidligere og nye erfaringer og betraktninger til å utvikle nye, mer spesifikke scener som kan benyttes i flere runder med eksperimenter.

Basert på disse betraktningene ble det igangsatt en prosess for å samle inn et stort og variert datagrunnlag tidlig i prosessen. Først og fremst handlet det om å samle inn data fra allerede eksisterende scener. Scenene som ble benyttet var i stor grad de samme som i det originale prosjektet; 'Jevn forflytning', 'Sakkadedeteksjon', 'Huskestue' og 'Drosjetur'. Samtidig ble samlet inn data for to ekstra eksperimenter, eksperiment 1- og 2. For å sørge for at vi hadde et tilstrekkelig stort og variert datagrunnlag ble det samlet inn data fra gruppens 3 medlemmer, samt 7 anonyme testpersoner. For scenene 'Drosjetur' og 'Jevn forflytning' ble det samtidig samlet inn data i to forskjellige

scenarioer, hvor testpersonene ble instruert til å nummer 1; bevege seg fritt i 3D rommet, og 2; holde hodet og kroppen i ro under hele varigheten av testen. Dette ble gjort for å få et inntrykk av påvirkningen på bevegelse av headsettet ville ha på støy i datasettet, noe som viste seg å være svært viktig for videre arbeid.



Figur 7: Hodet i bevegelse



Figur 8: Hodet i ro

Figur 7 og 8 illustrerer øyets bevegelse i x-retning (blå farge) og y-retning (oransje farge) over tid t i sekunder. Øyebevegelsen er samlet inn fra samme testperson, fra samme scene, hvor eneste forskjellen er hvorvidt testeren har blitt instruert til å holde hodet i ro under gjennomførelsen. Fra denne datainnsamlingen er det åpenbart at testene var avhengig av å i større grad kontrolleres for å forsikre at datainnsamlingen i så stor grad som mulig var egnet for å samle inn god og nøyaktig øyedata.

3.2.2.2 Kvantitativ evaluering av scener og algoritmer

Fra før var det gjennomført analyse ved hjelp av Cohens Kappa for å gjøre en kvantitativ evaluering av de forskjellige algoritmene målt opp mot manuelt klassifisert data. Etter samtaler med den Moan, Nygaard og Ramberg [63], ble det utrykt at evaluering ved hjelp av andre statistiske metoder. Samtidig ønsket vi å oppnå resultater og evalueringer som i så liten grad som mulig var avhengig av manuelt klassifisert data, på bakgrunn av den tidkrevende prosessen det har vist seg å være. Målsetningen med denne prosessen blir derfor ikke nødvendigvis å avgjøre riktighet, men heller å videre utforske hvorvidt man kan kartlegge brukbarhet, og begrense usikkerhet i design av ytterligere eksperimenter.

Friedmantesten er en metode som ikke har behov for fasit, men ser heller på relasjoner mellom de ulike scener og klassifiseringsmetoder vi benytter oss av [25]. Det ble derfor gjort en Friedmantest for både likheter mellom de eksisterende eksperimentene, og de implementerte algoritmene. Den første lar oss se om det eksisterer ikke-trivielle forskjeller i antall fikseringer og sakkader på tvers av scenene, dette gjelder også på tvers av scener med og uten forsøk på å redusere støyen. Den andre er for å se på forskjell i antall klassifikasjoner av sakkader og fikseringer per algoritme. Dette gjøres for å se om en kan vise forskjellige klassifikasjoner av samme scener for de enkelte algoritmene.

Det kan kanskje virke som trivielle øvelser, samtidig som det er vanskelig å skille hensikten av de to spesifikke anvendelsene av Friedmantestene. I praksis kan hensikten bak disse beskrives slik:

1. Vi ønsker å se om det er store forskjeller på hvordan øyebevegelser klassifiserer på tvers av scenene. Vi ønsker også tall på usikkerheten rundt de forskjellige klassifiseringene. Dersom forskjellene ikke er trivielle, kan dette gi konkrete tall som beskrives påvirkningen av forskjellige typer stimuli i scenene. Det kan også tallfeste påvirkningen av støy som følge av bevegelser i hodesettet.
2. Vi ønsker å se om det er store forskjeller i hvordan algoritmene klassifiserer den samme dataen. Ved store forskjeller kan dette fortelle oss om hvilke algoritmer som gir det mest generelle

bildet av øyedataen. Samtidig kan store avvik tyde på at enkelte algoritmer ikke egner seg i videre analyse. Her er det også nyttig å se hvorvidt en kan se om forskjellene potensielt er tilfeldige, eller om man konkludere med at usikkerheten er på innenfor akseptabelt nivå.

Å avgjøre hva som er akseptabelt nivå er usikkerhet kan være utfordrende. Gruppen setter ønsket konfidens til å være over 95 prosent, det er da nødvendig med en p-verdi lavere enn 0.001.

3.2.2.3 Utvikling av nye eksperimenter

På bakgrunn av betraktningene beskrevet i forrige delkapittel, sammen med samtalene med fagkyndige og erfarne personer innenfor feltet, ble det utviklet fire nye scener. Det første og viktigste aspektet med disse scenene var at den innsamlende dataen i så stor grad som mulig skulle beskrive øyets bevegelser som en konsekvens av designet av eksperimentet. Dette innebærer to spesifikke aspekter som ikke tidligere hadde blitt tatt høyde for ved tidligere eksperimenter. Den første var å begrense støy i dataen ved å i så stor grad som mulig eliminere bevegelser i headsettet. Den andre var at objektene som testpersonen skulle fiksere på under eksperimentet, hadde en posisjon i 3D rommet eksisterte relativ til rotasjonen til testpersonen og det tilhørende headsettet i 3D rommet. På denne måten vil objektene i rommet alltid være i samme posisjon i synsfeltet til testpersonen, uavhengig av testpersonens bevegelser og rotasjon i det faktiske rommet selv under eksperimentgjennomførelsen. Betydningen av dette i praksis er at uavhengig av hvor mye en person har beveget seg, så vil en i større grad kunne predikere den fremtidige blikkdataen for hvor en person kommer til å feste blikket, gitt at man vet posisjonen til objektet på forhånd. Dette er i motsetning til scener som 'Jevn forflytning', hvor bevegelser i 3D rommet og endringen av hodestilling vil gjøre at øyedataen blir mer uforutsigbar.

Samtaler med blant annet Ali Alsam, som tidligere har jobbet med nettopp innsamling av øyedata, forklarte oss at det kunne være hensiktsmessig å i større grad sammenligne scener på tvers av hverandre [64]. Tidligere har fokuset i stor grad omhandlet å klassifisere forekomst av øyebevegelser, hvor det ikke eksisterer annen fasit manuelle klassifiseringer. Det var derfor ønskelig å lage scener som i større grad kontrollerte testerens bevegelse gjennom testscenarioet, ved å gi testeren forskjellig type stimuli og frekvens på bevegelsene. Samtidig som testmiljøet var så likt som mulig på tvers av de forskjellige scenene. Formålet med disse scenene var å se om det var mulig å med høy grad av nøyaktighet å skille på type stimuli med å kun se på øyedataen. Samtidig vil det være mulig å gå inn for å se spesifikke tidsintervall hvor en med svært høy grad av sikkerhet kan si at en fiksering eller sakkade vil ha funnet sted. Dette eliminerer behovet for manuell klassifisert data til gruppens formål.

Testene ble designet med et grålig plan som sammen med objektene i scenen, beveger seg relativ til hodesettets posisjon. Objektene som skal fikseres på er grå sfærer som blir grønne ved kollisjon med blikkvektoren. Dette utgangspunktet ble benyttet til utviklingen av de fire unike scenene. Bevegelser og adferd kan endres og kontrolleres enkelt av de som benytter den. Det er mulig å randomisere testscenarioet med et 'seed', slik at en kan kjøre de samme testene samt sammenligne testresultater etter kjøringer. Hvor ofte objektet skifter posisjon, og farten de eventuelt beveger seg kan bestemmes av den som kjører eksperimentet. Dette kan også randomiseres ved behov.

3.2.2.4 Beskrivelse av scener

Den spesifikke implementasjonen av de fire forskjellige scenene er som følger:

One ball still inneholder en ball som oppdaterer posisjonen sin i rommet gitt randomisert eller bestemt intervall av tid. Hensikten med dette eksperimentet er å få testpersonen til å fiksere rolig på et objekt i rommet. Ved oppdatering av objektets posisjon vil det oppstå en sakkade frem til ballens nye posisjon i rommet, hvor en ny fiksering vil begynne.

One ball move inneholder en ball som oppdaterer posisjonen sin i rommet på samme måte som i One ball still. Forskjellen er at ballen her vil bevege seg langs x-aksen i en randomisert eller bestemt

hastighet avhengig av hva som bestemmes før gjennomføringen. Hensikten med dette eksperimentet er å få samme adferd som i One ball still, men hvor fikseringen skjer i en jevn forflytning.

Many ball still implementeres på samme måte som One ball still, men inneholder et randomisert antall mellom 7 og 14 baller. Hensikten med dette eksperimentet er å se om det er nevneverdig forskjell på øyedataen når en person får inn flere potensielle objekter å fiksere på.

Many ball disappear implementeres på samme måte som Many ball still. Denne er forskjellig ved at i det blikket til testpersonen møter et objekt i scenen, vil denne bytte posisjon til et annet sted i rommet. På denne måten vil ikke testpersonen ha et sted å feste blikket i rommet. Hensikten med denne er å se på øyedataen når testpersonen ikke har noen steder å fiksere på over lengre tid. Her er det forventet at det skal bli mange kortere fikseringspunkter, og svært mange ballistiske bevegelser i form av sakkader.

3.2.2.5 Eksperimentgjennomføring

Før datainnsamlingen ble testpersonene informert om prosedyre og hensikt med innsamlingen. De ble bedt om å signere et samtykkeskjema, hvor de også oppga eventuelle nær- eller langsynthet. All dataen fra uavhengige testpersoner ble randomisert, men tilhørigheten ble tatt vare på slik at de i ettertid kan slettes på forespørsel fra den aktuelle testeren.

Før et eksperiment gjennomføres må testdeltageren gjennomføre kalibreringen innebygget i Steam VR sin programvare. Deltageren blir bedt om å følge en ball som beveger seg i rommet. Kalibreringen tar også høyde for avstanden mellom brukeren sine øyne, og posisjonen på hodesettet relativt til brukerens øyne. Eksperimentvarigheten er 30 sekunder, og startes manuelt av testansvarlig så fort testdeltager sier seg klar til å begynne testen. Ved ferdig gjennomført test blir øyedataen automatisk lagret og navngitt med deltager og seed, og plassert i tilhørende mappe. Under eksperimentene ble testpersoner instruert til å sitte helt i ro, og kun bevege øynene sine gjennom varigheten på testen. Scenene ble kjørt med tilfeldig oppdateringsfrekvens på objektene sin posisjon, og i scenen hvor ballens fart var en faktor ble denne også randomisert. Det ble samlet inn data med fire forskjellige seeds.

I tillegg til gruppens tre medlemmer, ble det samlet inn data fra syv andre deltagere. Datagrunnlaget for disse 30 sekunder lange eksperimentene var et datagrunnlag på omkring 80MB med øyedata fra totalt 78 unike datasett. For å ha et tilstrekkelig datagrunnlag for maskinlæring ble de samme eksperimentene gjennomført på gruppens medlemmer nok en gang med et nytt tilfeldig seed. Denne gangen ble eksperimentene kjørt 270 sekunder hver, som ga nye 120 MB med data fordelt på de tre scenene. Den totale mengden data i det nye datagrunnlaget fra scenene gruppen utviklet var derfor omkring 200MB. Dette tilsvarer over en time med øyedata. Ved siste kjøring av de endelige testene for *RNN-modellene*, ble det samlet inn ytterligere 120 MB med 270 sekunder lange datasett. Det totale datasettet ved prosjektets slutt ble totalt 313 MB med øyedata.

3.2.3 Databehandling

Fra øyedataen kan vi regne ut vinkelhastigheten og vinkelakselerasjonen til øyet. Den temporale spredningen kan også regnes ut fra denne dataen. Som presentert i Moan, Nygaard og Ramberg sitt arbeid kan disse parameterne benyttes for å klassifisere øyebevegelsene på et gitt punkt i dataen, ved hjelp av forskjellige okulomotoriske algoritmer. For å gjøre en tilstrekkelig vurdering av algoritmene i seg selv, og konklusjonene som er trukket i det tidligere arbeidet, er vi avhengig av en systematisk metode for databehandling. Etter data er samlet inn, kjøres `classification_handler`, som går gjennom valgte øyedatafiler og klassifiserer vært punkt med hver eneste algoritme. Samtidig regnes det ut en mean average, samt den faktiske gjennomsnittlige klassifiseringen. Klassifiseringen lagres i en egen fil for å illustrere forskjellene i klassifiseringer på tvers av algoritmene. Dette gir mulighetene til å analysere dataen på spesielt tre aspekter.

1. Det er mulig å se overordnende forskjeller på klassifikasjoner på tvers av algoritmene, basert på antallet klassifiserte sakkader og fikseringer for hver algoritme.

2. Det er mulig å se forskjeller i klassifiseringer for spesifikke punkter i dataen.
3. Det er mulig å se forekomsten av totalt antall klassifiserte bevegelser for de forskjellige scenene

De spesifikke feltene som lagres direkte i Unity under prosjektgjennomførelsen beskrives på Gitlabsiden. Etter klassifiseringen i Python får vi også disse nye feltene som kan brukes i analysen.

dAngle	Vinkel mellom sist målte blikkvektor og nåværende
dt	Delta tid fra forrige datainnhenitng
t	Tid I sekunder fra start av scene
dAdt	Vinkelakselerasjon
dVdt	Vinkelhastighet
disperision	Romlig spredning

3.2.3.1 Vinkelhastighet og akselerasjon

For å regne ut vinkelhastighet må vi først regne ut vinkelendring mellom påfølgende blikkvektorer. Dette gjøres ved hjelp av følgende formel:

$$\theta = \arccos \frac{\vec{v} \cdot \vec{u}}{|\vec{v}| \times |\vec{u}|}$$

Deretter regner vi ut vinkelhastighet ved å dele på tidsforskjell mellom datapunktene. For å regne ut vinkelakselerasjon ser man på endring i vinkelfarten mellom to påfølgende datapunkter og deler dette igjen på tid.

3.2.3.2 Romlig spredning

Romlig spredning (eng.: dispersion) kan beregnes på flere måter avhengig av den spesifikke implementasjonen en ønsker å benytte. En trenger først å fremst å definere vinduet en ønsker å regne ut spredningen til. Den romlige spredningen er avstanden, målt i grader, mellom de to vektorene som ligger lengst fra hverandre i vinduet [36].

3.2.3.3 Støyreduksjonsmetoder

Vi valgte å implementere Moving Median, Mooving Average og Tobii sin egen støyfilter algoritme [45]. Algoritmene benytter er glidende vindu med et gitt antall punkter hvor ulike funksjoner blir kjørt for å finne en ny verdi. Moving average benytter seg av gjennomsnitt og Moving median bruker median verdi i vinduet. Tobii filteret kjører gjennomsnitt av et gitt antall punkter før og etter seperat, før det gjøres et nytt gjennomsnitt av gjennomsnittene.

Vi valgte å likevel ikke å benytte oss av filtrene ettersom resultatene ikke var tilfredstillende. Dette står beskrevet *senere i rapporten.*

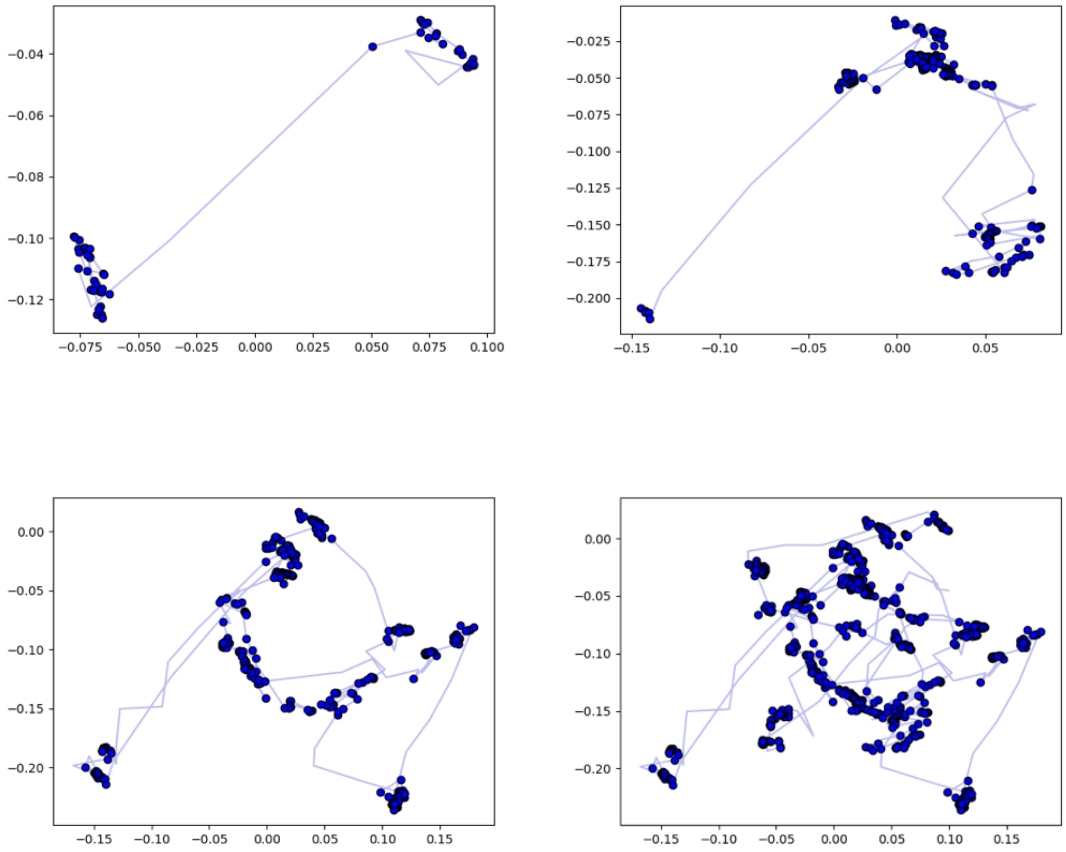
3.2.4 Konvolusjonelt Nevralt Nettverk

3.2.4.1 Opprettelse av datasett

For å trene opp et konvolusjonelt nevralt nettverk ble det opprettet bilder av øyedataen for forskjellige tidsintervaller. Tidsintervallene var henholdsvis et, tre, fem og ti sekunder lange. Disse intervallene ble valgt for å lage bilder med ulik lengde og samtidig som forskjellene mellom dem ville være distinkte nok til å skille. Intervallene passet med utformingen av de nye scenene hvor ballene flyttet seg med opp til to sekunders mellomrom. Dette fører dermed til at bildene inneholder

ulikt antall oppdateringer for ballene. Med intervaller over ti sekunder, vil uansett datagrunnlaget bli for lite for bruk i trening.

Figur 9: 1, 3, 5 og 10 sekunders øyedata



Ingen av datapunktene benyttes i to forskjellige bilder, dette betyr at antall bilder tilgjengelig for trening, validering og test avhenger av varigheten på tidsintervallet for hvert bilde. Fordelingen på disse var 60 prosent til trening, 20 prosent til validering, 20 prosent til validering. Det totale datagrunnlaget på bakgrunn av timen med øyedata tilgjengelig for de forskjellige tidsintervallene blir følgende:

Tabell 1: Datagrunnlag

Tidsintervall	1	3	5	10
Antall bilder	4 965	1 605	1 005	429
Trening	2979	963	603	258
Validering	933	321	201	86
Test	933	933	321	85
Størrelse mb	132	57	44	25

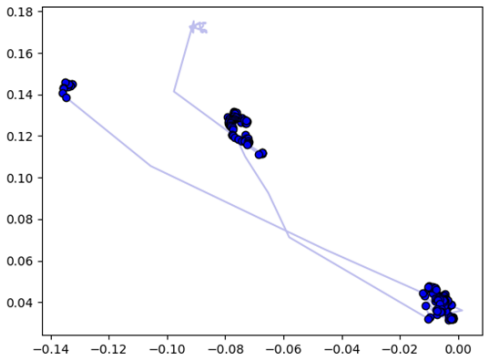
3.2.4.2 Forarbeid

Ved opprettelse eller oppdatering av data for bruk i trening og testes, må det lages nye bilder. Dette gjøres automatisk i `create_picture.py` hvor dataen lagres og sorteres, og tilhørende fasit automatisk skrives. Programmet `split_data.py` randomiserer og splitter datasettet opp i ønsket fordeling trening, test- og valideringssett, i gruppens tilfelle henholdsvis 80, 20 og 20 prosent.

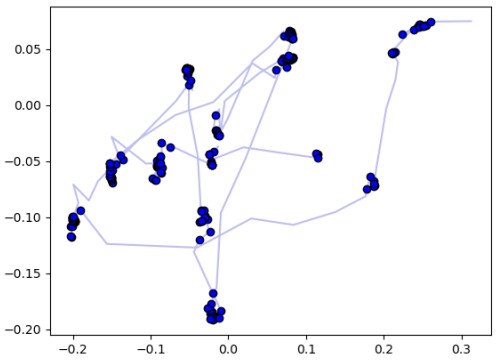
Programmet `preprocess.py` som benytter OpenCV leser inn bildene, klipper bildet slik at det kun inneholder øyedataen, og bruker Pickle til å lagre dataen som binærfiler for bruk i modellen.

Hvert bilde registreres med sin tilhørende klasse, som illustrert.

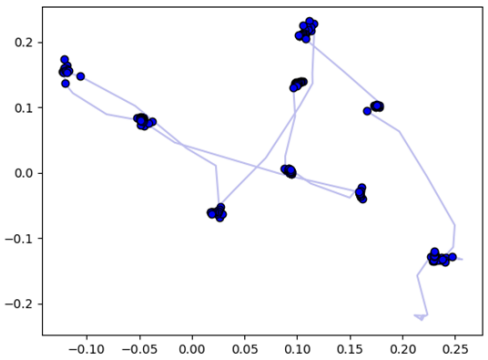
Figur 10: One ball still, 3 sekunder



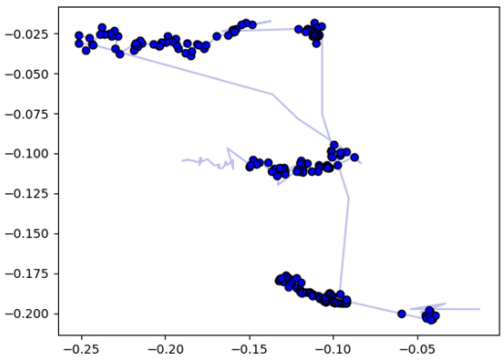
Figur 11: Many ball disappear, 3 sekunder



Figur 12: Many ball still, 3 sekunder



Figur 13: One ball move, 3 sekunder



Klassefordeling CNN

Klasse 0	One ball still
Klasse 1	Many ball disappear
Klasse 2	Many ball still
Klasse 3	One ball move

Alle bildene stokkes om og fordeles tilfeldig på trening, validering og testsettet. Dette gjøres for å sørge for at datasettene er uavhengige av hverandre, da spesielt med tanke på testsettet. En av konsekvensene av dette er at det blir noen ulikheter mellom antall forekomster av hver klasse. På generelt grunnlag er fordelingene av de forskjellige scenene ganske jevne for hver kjøring, med noen ulikheter. Disse beskrives på følgende måte.

Tabell 2: Datagrunnlag

Klasse	0	1	2	3
1 sek	766	756	738	719
3 sek	262	212	239	250
5 sek	141	153	165	144
10 sek	67	64	58	70

3.2.5 Recurrent Nueral Network

Recurrent Nueral Networks (RNN) brukes ofte i Natural Language Processing. I stedet for å ta inn en stor samling med datapunkter, så kan en RNN ta inn et og et datapunkt som input. Dette tillater større grad av fleksibilitet for lengden av input. Dersom man tar inn mange datapunkter, for å klassifisere én enkelt scene blir RNN-arkitekturen 'mange til en'. Den største forskjellen fra vanlige NLP-problemer og å bruke RNN-med øyedata, er at vi bruker numerisk data hvor differansen i verdier må bli tatt høyde for. Denne spesifikke implementasjonsdetaljen skiller dette prosjektet fra typiske NLP-problemer.

3.2.5.1 Forarbeid

For å kunne benytte seg av den innleste øyedataen, er det nødvendig å tilpasse modellen til å benytte seg av numeriske verdien, i motsetning bokstaver og karakterer slik som i mer vanlige NLP-problemer. I tidligere arbeid opprettet vi en 'dictionatry' som ga hver ulik bokstav en numerisk verdi, såkalt 'character embedding' [10]. I denne oppgaven anså vi det som nyttig å gi flere felter som input. Ettersom verdiene vi valgte å benytte oss av allerede var numeriske var det derimot ikke et behov for embedding. Vi la derimot opp til at modellen kunne klassifisere flere klasser samtidig (scener og personer). Dette ble gjort ved å benytte numerisk liste for merkelapper på data. Der avgjør 0 eller 1 på den respektive plassen om data tilhører en klasse eller ikke.

3.2.5.2 Minibatches

Ettersom vi samlet inn store mengder data var det ikke hensiktsmessig for grafikkortet å holde på hele datasettet samtidig. For å begrense minnebruken under trening blir datasettet dermed delt inn i mindre batches som sekvensielt sendes til grafikkortet. Dette fører til at grafikkortet kan konsentrere seg om en mindre mengde data, en batch, av gangen.

En sekvens med øyedataen må være like lang for å kunne sendes inn i en batch. I prosessen for å oppnå dette deles først data opp i sekvenser av ulik lengde og blir merket med de rette klassene. Deretter sorteres data etter lengde slik at hver batch er tilnærmet lik sekvenslengde. Til slutt paddes (legges til 0-ere) sekvensene opp til største sekvenslengde innad i batchen. Padding taes hensyn til når input sendes inn i modellen, slik at dette ikke skal påvirke klassifiseringen. For at modellen skal ha informasjon om dette sendes upaddet sekvenslengder for hele batchen sammen med input.

3.2.5.3 Bidirectional

Det er mulig å benytte to skjulte lag i LSTM, med motsatt retning, til samme utgang. Det dette i praksis betyr er at punkter kan få kunnskap om andre punkter i begge retninger. Ved overføring av modellen til ONNX og barracuda fant vi derimot ut at dette ikke var støttet. Av den grunn valgte vi å trene modellen uten denne egenskapen, men gjorde det enkelt å implementere ved eventuelle fremtidige oppdateringer med utvidet støtte. For anvendelser uten behov for sanntidsdeteksjon, ansees dette som nyttig å implementere for økt ytelse på modellen.

3.2.5.4 K-Fold

Når man benytter seg av et mindre datasett kan det være vanskelig å vite om resultatene man får under trening er på grunn av tilfeldigheter. K-Fold er en teknikk der man trener opp en modell fra start av K ganger. Hver gang blir en ulik del av datasettet tilsidesatt og benyttet som valideringsdata. Dette gjør at dersom man får lignende resultat ved de ulike treningene av modellen, kan en i høyere grad stole på resultatet [60]. Dette ble implementert ved trening av LSTM modellen vår. Modellen ble trent opp 5 separate ganger og 1/5 av treningsdata tilsidesatt ved hver kjøring. Etter kjøring testet vi de ulike modellene på et uavhengig testsett og regnet ut

gjennomsnittlig presisjon og standardavvik. Dette ga et tall på prestasjon og hvor mye variasjon det var på tvers av modellene.

3.2.5.5 Datainndeling

For å benytte seg av dataen på en hensiktsmessig måte, er det viktig å ta informerte valg rundt datainndeling knyttet opp mot trening, validering og testing. For RNN-modellen blir anvendelsen litt annerledes enn for CNN-modellen, og som konsekvens av dette er det altså enkelte nye aspekter som må hensyntas. Under utvikling og videreutvikling av modellene, ble det behov for enda mer data enn det tidligere hadde blitt samlet inn. Derfor ble det samlet inn ytterligere fem minutter med øyedata fra de fire scenene, for gruppens tre medlemmer. Dette ble gjort med et nytt uavhengig randomisert seed. Dette ga et datagrunnlag på litt over to timer med øyedata fordelt på de ulike seedene. For testingen ble det benyttet helt uavhengig datasett fra ulike kjøringer enn de som ble trent på. Her ble det samlet to datasett per scene for gruppens tre deltagere. Et av disse var med random seed 4 som også ble benyttet i treningen, et annet var med seed 102, som ikke ble benyttet under treningen. På denne måten var det også mulig for gruppen å teste hvorvidt eventuelle gode treningsresultater kom av at den lærte seg mønsteret på oppdatering og objektene i scenen, eller om det også ville fungere på et helt uavhengig testsett.

3.2.5.6 Sekvenslengder

Et viktig spørsmål når en skal trene på sekvensiell data i LSTM, er hvor sto input lengde som er hensiktsmessig å trene på, samt bruke under klassifisering. LSTM er svært anvendbart til blant annet identifikasjon av språk, her vil det naturligvis være ulik inputlengde avhengig av hvor lange setninger eller avsnitt man benytter. Disse kan ha stor variasjon. Noe av det samme kan bli sagt om øyedata, dette har vi derimot mulighet til å kontrollere i større grad. Hvor mye en kan bruke til klassifikasjon avhenger av hvor mye data man skal klassifisere, og eventuelt hvor ofte en ønsker å få tilbakemelding fra modellen i sanntid. I dette tilfelle er det svært ønskelig at modellen i hvert fall skal kunne klassifisere med god presisjon uten behov for stort mer enn noen få sekunder. Dette vil gjør en modell mer skalerbar for andre typer anvendelser enn kun klassifikasjon av scener.

Trening av modellen ble forsøkt med forskjellig tilnærming til behandling av inputdata. Den ble trent på både en fast inputlengde, samt variabel inputlengde. Ved trening med variabel input lengde, ble det delt opp i forskjellige sekvenser med øyedata av likt antall datapunkter i hver sekvens. Det ble også testet med variabel lengde på sekvensen med øyedata etter trening. Ved å dele testdataen opp i forskjellige sekvenser, kan en si noe om hvor mange datapunkter som er hensiktsmessig å benytte seg iblant annet sanntidsklassifisering.

3.2.5.7 Valg av hyperparametere

I sitt verk beskrev S. Sims, V. Putnam og C. Conati et behov for å utforske hyperparameteres påvirkning for å optimalisere treningen på øyedata med RNN [52]. Før de endelige resultatene til modellen presenteres, er det derfor viktig å vise benyttede hyperparameteres påvirkning på modellens brukbarhet.

Valg av hidden size og treningsdetaljer

Gruppen har tidligere jobbet med LSTM, hvor det viste seg at endring i Hidden Size kunne ha relativt stor påvirkning på den endelige ytelsen til en maskinlæringsmodell. Samtidig vet vi at økt hidden size øker treningstiden markant. Modellen ble testet ut med hidden size: 128, 256, 512 og 1024, tilsvarende $2^7, 2^8, 2^9 og 2^{10}$.

Hidden Size 512 ble fra testene vist å ha gode resultater for presisjon og loss, samtidig som at treningstiden ikke ble for lang. Den ble derfor benyttet i videre trening.

Variabel læringssrate

For å optimalisere presisjon og tap implementeres en læringsplanlegger som gjør at treningen skjer med variabel læringsrate. Læringsraten har en startverdi 0.001, og hver sjette epoke så deles denne verdien på ti. Dette gjøres slik at modellen i starten skal ha fokus på å utforske forskjellige muligheter i treningen, mens den senere i treningen skal ha større fokus på optimalisere det den har lært.

3.2.6 Klassifisering i sanntid

Etter ferdig trening og evaluering av de ulike LSTM modellene var målet videre å benytte disse i sanntid i Unity. Det var ønskelig å eksportere modellen til Unity for sanntidsklassifisering av spesielt to grunner. For det første kan svar i sanntid være nyttig, da det vil gi mye raskere tilbakemeldinger enn å kjøre datasettet gjennom de forskjellige prosesseringsverktøyene i Python prosjektet. Den andre praktiske bruken er mer forskningsrettet. Å behandle store data er både tidkrevende og til tider vanskelig. Selv med informerte og gjennomtenkte valg, er det fortsatt en mulighet for feil i prosessen, som fører til unøyaktige resultater. Dersom modellen også klarer å klassifisere dataen med høy grad av presisjon i sanntid, eliminerer man mye av usikkerheten knyttet til datainndelingen, treningen og testingen.

For å gjøre dette må modellene først eksporteres til et format som kan benyttes, ettersom programmeringsspråket i Unity er C#. Vi benyttet ONNX formatet og installerte en tilleggspakke i Unity kalt Barracuda, som støttet avlesing [57]. Videre lagde vi et klassifikasjonsskript som var mulig å benytte i de ulike scenene. I klassifikasjonsskriptet benyttet vi en NNModel og IWorker som er del av Barracuda API-et. En NNModel leser av en valgt modell og IWorker klassifikasjoner ved å kjøre input gjennom modellen. Input hentes fra rå øyedata ved kjøring av scenene. Før input sendes inn i modellen blir eventuelle felter, som ikke er del av rådataen, regnet ut. I vår implementasjon av modellene benyttet vi blant annet fart, og som beregnes ut ifra øyets blikkvektor mellom to påfølgende punkter. I tillegg blir data omformet til et riktig format for å kunne benytte det i modellen. Skriptet henter inn øyedata i 3 sekunder av gangen. Dette tilsvarer ved normal kjøring mellom 160-240 datapunkter, ved benyttelse av skriptet. For hvert intervall predikerer modellen hvilken scene og/eller hvilken person som kjører, ut ifra hvilken modelltype som er implementert.

3.2.7 Time Series Forecasting

Interessen for forecasting kom på bakgrunn av et ønske for å kunne predikere øyets fremtidig posisjon, basert på forgående øyebevegelse. En forecasting løsning vil tillate å se på den tidligere dataen i et datasett opp til et punkt, og bruke den informasjonen som grunnlag for å predikere hva fremtidige datapunkter vil være. Det umiddelbare problemet som oppstår i sammenheng med fremtidig prediksjon er mangelen på rytme og fast struktur i øyebevegelser over tid. Ettersom øyebevegelser kan utspille seg på bakgrunn av ytre stimuli eller en persons egen vilje er det vanskelig å bare belage seg på tidligere data for å komme med generelle prediksjoner om fremtidig øyedata. Det finnes likevel tilfeller av øyebevegelser hvor det lar seg gjøre. Sakkader sin ballistiske natur gir bevegelsen den fastsatte og klare formen som forecasting trenger for å kunne utføre prediksjoner. Modellen bruk fra Darts biblioteket var N-BEATS modell. Neural Basis Expansion Analysis for Interpretable Time Series Forecasting (N-BEATS) er en dyp nevral arkitektur modell som ble først presentert av Oreshkin, Carpov, Chapados og Bengio[13].

3.2.7.1 Darts

Det finnes ikke en bestemt måte å løse et forecasting problem. Basert på problemet man står ovenfor kan det være en rekke mulige måter å løse forecastingen på. Man kan belage seg på Regresjon, nevrale nettverk eller 'Random forest' bare for å nevne noen eksempler [28]. Bruken av Darts biblioteket til Python åpnet for flere unike løsninger. Biblioteket gjorde det mulig å utforme en modell som trente ikke bare på et datasett, men tok heller inn en rekke datasett, hvor hvert datasett var en enkelt sakkade. Dette ble implementert ettersom forecasting belager seg på et sliding window prinsipp når den predikerer det neste datapunktet i et datasett. Et datasett hvor

man bare hadde en rekke sakkader etter hverandre ville bydd på problemer da det glidende vinduet ville benyttet seg av de avsluttende punktene fra en sakkade til å forecaste sakkaden som kom neste i datasettet. Darts gjør det også enkelt å gjøre modellen multivariate. Dette vil si at forecasting modellen vil ta høyde for en rekke features i datasettet, ikke bare en. Dette var sentralt ettersom features som blikkets gaze vektorer, vinkelhastighet og akselerasjon alle kan være med på å fortelle hvor sakkaden vil bevege seg videre. I tillegg gjorde Darts det lettvint å håndtere modellen som en multi-target, altså at modellen ikke bare forecastet en feature, men flere features.

3.2.7.2 Datasett

Scenen one ball still som tidligere beskrevet i punkt 3.2.1.1 ble brukt til innsamling av data til et datasett. Forflytningsintervallet på ballen ble satt til 1 sekund, og scenen ble kjørt i 5 minutter. Dette ga et omfattende datasett med en fast rytme og flere sakkader. Dette datasettet ble deretter klassifisert med HMM algoritmen tidligere implementert av Moan, Nygaard og Ramberg. Basert på denne klassifiseringen ville datasettet bli delt opp i flere datasett, ett for hver sakkade. Disse datasettene blir delt opp under kjøring, og eksisterer bare i minne for å forhindre å måtte lagre utallige mindre datasett for hver kjøring man ønsker å gjennomføre. Hvert datapunkt i et datasett inneholder en rekke features, ikke alle like nyttig for forecastingen. Det ble derfor utført en feature selection hvor de featurene som har stort utslag for påvirkningen av blikkets bevegelse ble tatt med i treningssettet. Disse featurene er X, Y og Z vektorene for blikket, vinkelhastighet samt vinkelakselerasjon.

3.3 Arbeids- og rollefordeling

Gruppens tre medlemmer er likestilt, og er alle sentrale i løsningen av problemet. Som følge av at gruppen er så liten som den er, ble det ikke vurdert som hensiktsmessig å dele inn i tydelige roller som er typisk for smidig utviklingsmetodikk. Gruppemedlemmene har ganske lik fagbakgrunn for relevante emner knyttet opp mot dette bachelorprosjektet, og alle medlemmene har vært delaktige i majoriteten av de ulike delmålene. Likevel ble ansett som nyttig å delegere hovedansvar for oppgaver til enkeltmedlemmer, spesielt i oppstartsfasen av den spesifikke deloppgaven. På denne måten kunne et medlem gå i dybden og skaffe teoretisk innsikt etter behov, før eventuelle andre medlemmer bisto med hjelp og nye perspektiver. Overordnet kan de store arbeidsoppgavene etter hovedansvar beskrives som følgende:

1. Stian Fjæran Mogen: Utvikling av scener og konvolusjonelt nevralt nettverk
2. Simon Jensen: Recurrent nevralt nettverk
3. Lars Brodin Østby: Forcasting

Det skal likevel presiseres at gruppen spilte på hverandres styrker, og delegerte oppgaver i felleskap etter behov, og alle gruppemedlemmene hadde et likt antall arbeidstimer. Selv om ikke all teorien i smidig utviklingsmetodikk var anvendbart for dette prosjektet, er det likevel noen prinsipper som har blitt benyttet. På starten av hver dag hvor gruppen var samlet, tok medlemmene en kjapp gjennomgang av hvilke arbeidsoppgaver en hadde planlagt å jobbe med fremover, og om en så for seg at en ville trenge hjelp. Gruppen benyttet seg også av oppslagstavlen Trello, da spesielt i faser av arbeidet hvor det var endringer i fokus. Dette kan eksempelvis være ved planlegging av nye scener, eller ved distribuering av delkapitler på hovedrapporten. Teknologien sin hovedoppgave var å visualisere og gi en oversikt over arbeid i perioder hvor det var usikkerhet hva en skulle jobbe videre med. I et prosjekt som dette er det viktig at samtlige av gruppens medlemmer kjenner at de til enhver tid har en retning å følge, og en arbeidsoppgave å holde på med. Samtidig er det helt essensielt at gruppemedlemmene kjenner mestring og ikke står alene i vanskelige problemstillinger over lang tid. Gruppens medlemmer ble derfor oppfordret til å be om hjelp dersom en hadde behov for dette.

Det var også et tema for gruppens medlemmer, at det var viktig at alle medlemmene hadde oversikt over hva som var gjort. Som følge av at det var et forskningsspørsmål, kunne det ganske raskt være

behov for å endre og spisse inn fokuset. Sånn sett var gruppens tilnærming at det var viktig at alle gruppemedlemmene hadde tilstrekkelig kompetanse om de forskjellige oppgavene, selv de en ikke hadde som fokusområde selv. Dette for å sørge for en tilstrekkelig teoretisk innsikt om forskningsarbeidet, samt god oversikt over prosjektarbeidet som helhet.

4 Resultater

4.1 Vitenskapelige resultater

4.1.1 Datasett

Etter eksperimentene var gjennomført var det mulig å gjennomføre analyse for å avgjøre hvordan dataen kunne benyttes på best mulig måte. Alle datasett av de forskjellige scenene fra den initiale datainnhentingen, ble klassifisert med samtlige algoritmer. På denne måten var det mulig å se på hvor ofte de forskjellige algoritmene klassifiserte sakkader og fikseringen. Resultatene ga grunnlag for å gjennomføre en Friedman test for å se på forskjellene mellom klassifikasjonsresultatene. Dette har tidligere blitt vist som en egnet måte å analysere lignende testresultater for øyedata av G. G. Dalveren og N. E. Caglitay [35]. En måte å bruke denne dataen på er å se om det er store forskjeller på tvers av scenene.

Tabell 3: Friedmantest på scener

Scene	Sum	Snitt
Smooth-Movement-Move	12	1.7
Smooth-Movement-Still	22	3.1
Taxi-Tour-Move	40	5.7
Taxi-Tour-Still	21	3.0
Eye data 1	25	3.6
Eye data 3	44	6.3
Look and Remember	32	4.6

Resultater fra Friedmantesten med $k = 7$ scener og $n = 7$ algoritmer

Høyere sum og snitt betyr flere klassifiserte sakkader

Friedmantesten gir en score på antall klassifiserte sakkader i forhold til fikseringer, for hver gitte scene. Deretter får de en plassering relativ til de andre scenene. Noen av scenene fra første datainnsamling ble kjørt lengre enn andre. Taxitour kunne bli kjørt opp mot et minutt, smooth movement kunne være rundt 30 sekunder. Det ble derfor benyttet 30 sekunder fra vært datasett. Datasettene ble justert til å ha samme antall datapunkter som experimentet med flest registrerte sakkader og fikseringer sammenlagt, slik at summen ble et representativt mål av forholdet mellom sakkader og fikseringer. Fra tabell 3 leser vi at eye data 3 og taxi-tour-move gir flest sakkader. Smooth-movement-move klassifiserer færrest. Resultatet fra friedmantesten er en friedmanverdi på 23.45, som viser store forskjeller. P-verdien er på 0.00066, som gir en konfidens på langt over 95 prosent ($p = 0.005$), slik at vi kan forkaste hypotesen om at forskjellene kommer fra tilfeldigheter.

Ettersom disse scenen i stor grad er designet for å fremprovosere forskjellige typer øyebevegelser er det ingen overraskelse at det er store, ikke trivielle variasjoner mellom de forskjellige scenene. Det som derimot er svært interessant er å se på de store forskjellene på samme scener, avhengig av om hodet er i ro, eller om hodet beveger seg fritt. Spesielt på smooth movement, hvor den eneste forskjellen i scenene er bevegelsen på headsettet. Her blir det klassifisert betraktelig flere sakkader enn tilsvarende scene med hodet i ro. Her er utgangspunktet at blikket til enhver tid skal fiksere på samme objekt gjennom hele kjøringen. Resultatet tyder på at påvirkningen fra støy er stor selv der en kan forvente svært like resultater. Å få tallfestet påvirkningen av støy på øyedataen, bekrefter behovet for videreutvikling av scener.

En vinkling som også vil være interessant for gruppens tilfelle, er å se hvor stor grad det er forskjeller på tvers av algoritmene, og hvilke algoritmer som gir den mest generelle klassifiseringen på tvers av disse. Her er det ikke å nødvendig å justere for scenenes lengde, da hver algoritme vil klassifisere på hvert datasett, og derfor likt antall datapunkter.

Tabell 4: Friedmantest på scener

Type	Sum	Snitt
I-DT	299	3.9
I-VT	96.5	1.3
NH	157	2.1
HMM	522	6.8
IMST	344	4.5
IMST 2	406	5.3
FIR	303.5	4.0

Resultater fra Friedmantesten med k = 7 algoritmer og n = 76 datasett
Høyere sum og snitt betyr flere klassifiserte sakkader

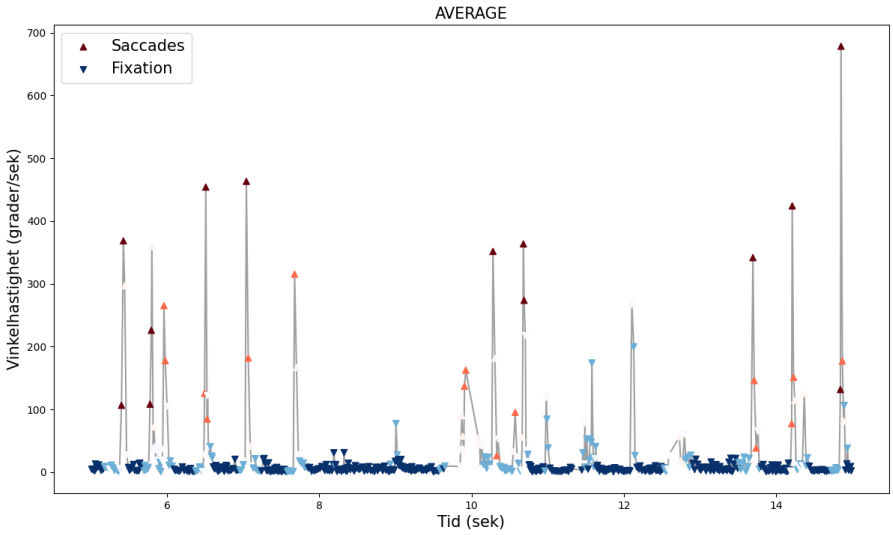
Friedmantesten gir her en mean average score basert på hvilken plassering algoritmene får i klassifikasjonene for hver av scenene. IVT er naturligvis den algoritmen som klassifiserer minst sakkader, og har dermed en betraktelig lavere score enn resten av algoritmene. NH algoritmen klassifiserer også relativt få, men denne vil også klassifisere andre typer øyebevegelser. HMM er algoritmen som klassifiserer klart flest sakkader, etterfulgt av IMST algoritmene. Et sted mellom ligger FIR og IDT, som klassifiserer en mer moderat mengde sakkader. Antall datasett som ble brukt i analysen var n = 76, med k = 7 forskjellige algoritmer. Dette gir en F-verdi på 350.24, som forteller oss at det er svært stor ikke triviell variasjon på tvers av disse syv algoritmene. Verdien for usikkerhet $p = 1.37^{72}$. Denne svært lave verdien kommer som følge av det store datagrunnlaget, og er nok en gang langt under grenseverdien 0.001 for 95 prosent konfidens.

Det er stor variasjon i hvordan de forskjellige algoritmene velger å klassifisere øyebevegelser, selv de som tar høyde for de samme parameterne. På bakgrunn av analysen kan vi ikke vurdere hvilken algortime som er best egnet til å klassifisere sakkader og fikseringer. Vi kan derimot vurdere hvilke algoritmer som skiller seg ut, og hvilke som gir mer gjennomsnittlige resultater.

4.1.1.1 Usikkerhet i klassifiseringen

På bakgrunn av de store ulikhetene i klassifiseringer av antall sakkader og fikseringer, dukket det opp et behov for å visualisere i hvilke punkter algoritmene var uenige. I tillegg til algoritmenes egne klassifikasjoner ble det nå også regnet ut en mean average og en faktsik gjennomsnittlig verdi. I dette tilfellet ble forenklingen gjort slik at en fiksering har verdi 1, og en type sakkade har verdi 2. På denne måten vil alltid den gjennomsnittlige verdien ligge mellom 1 og 2, mens mean average vil være en verdi som enten er 1 eller 2. På denne måten kan man se nøyaktig hvilke punkter algoritmene er uenige.

Figur 14: Usikkerhet i klassifisering

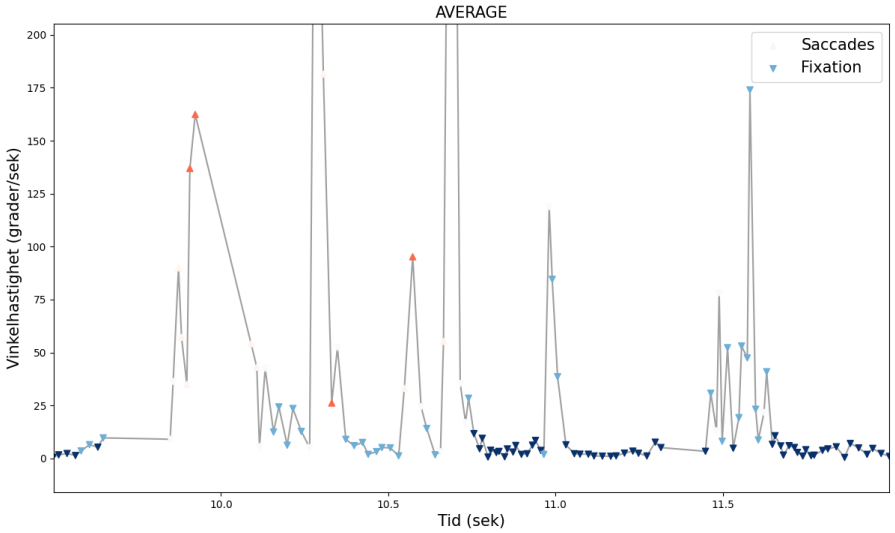


Sakkader i rødt, fikseringer i blått

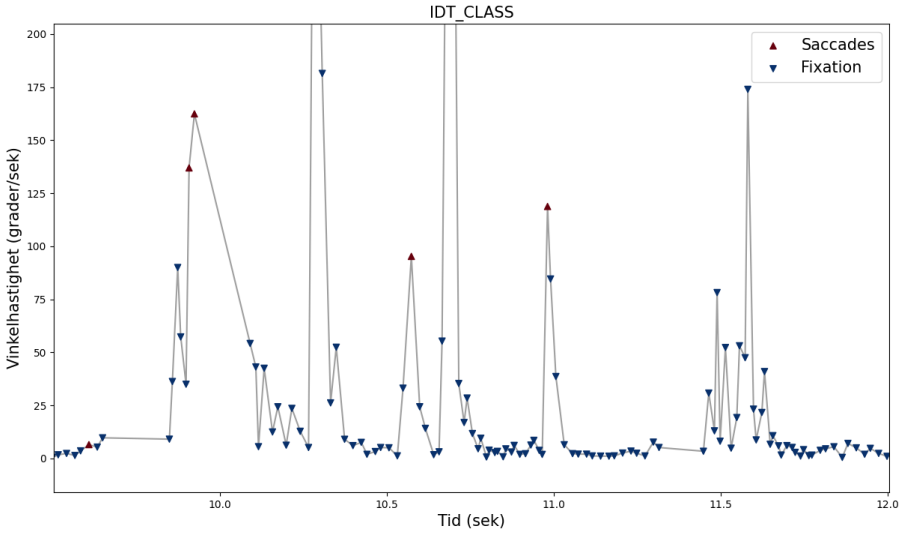
Sakkader i rødt, fikseringer i blått. Desto mørkere farger, desto mer enighet.

I figur 14 er det markert der sakkader og fikseringer med henholdsvis røde og blåe punkter, av ulik grad basert på hvor stor usikkerhet som er knyttet opp mot klassifiseringene fra algoritmene. Desto sterkere/mørkere fargen er, desto mer enighet er det på tvers av algoritmene. Puntkene som er hvite er de hvor algoritmene er mest usikre, og ikke klarer å skille mellom klassifiseringene. Algoritmene er mest enige om sakkader på punkter med høyest vinkelhastighet. De er mest enige om fikseringer på punkter med lav vinkelhastighet over en lengre tidsperiode. Uenighetene ser vi spesielt i områder før og etter høy vinkelhastighet Dette er punkter hvor algoritmer som IVT velger og si fiksering fremfor sakkade, mens algoritmer som tar høyde for romlig spredning kan klassifisere en sakkade.

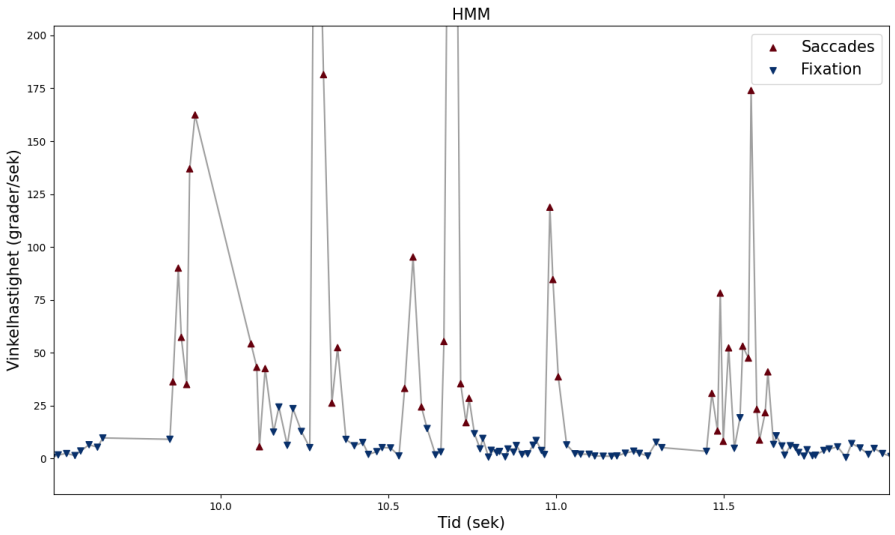
Teorien som er presentert sier viser at det er ulikheter i algoritmer basert på om de hensyntar den romlige spredningen i klassifiseringen. Resultatene fra Friedmantesten indikerer at det også er svært store forskjeller på algoritmene slik de er implementert. Også figur 14 viser dette, da med spesielt mye uenighet rundt sekund 10.



Figur 15: Usikkerhet i klassifisering



Figur 16: IDT for Taxi-Tour-Still



Figur 17: HMM for Taxi-Tour-Still

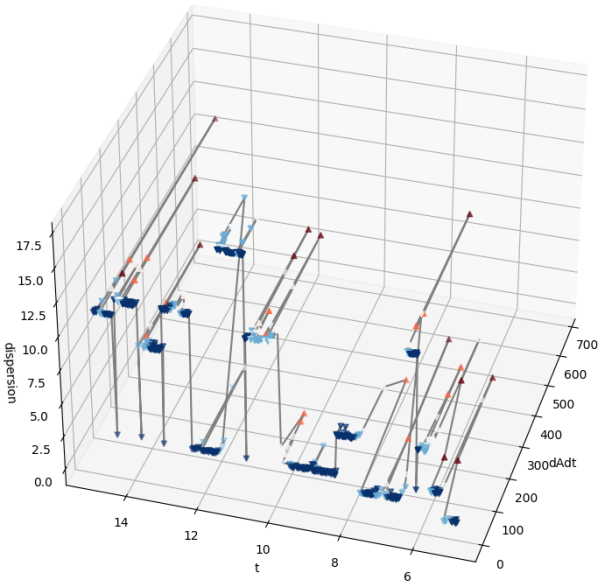
Figur 14 viser gjennomsnittlig klassifisering på tvers av algoritmene. Figur 16 og 17 viser klassifisering av samme datasett for henholdsvis I-DT og IHMM. Selv for algoritmer som i utgangspunktet har som hensikt å klassifisere sakkader basert på de samme parameterne, er det fortsatt stor grad av uenighet. Det er åpenbart at kun å sammenligne klassifisere scener med vinkelfart ikke er nok for å få en forståelse for øyebevegelser i en scene.

4.1.1.2 Datavisualisering

Etter samtaler med Ali Alsam, ble det åpenbart at det var et behov for forbedring og videreutvikling av hvordan dataen visualiseres, for å ta en grundigere vurdering på hvordan algoritmene klassifiserer dataen [64]. Tidligere har dataen i stor grad blitt visualisert i to dimensjoner, med fart og tid som eneste parametere. Spesielt i spørsmål om manuell klassifisering er dette mangelfullt. Både romlig spredning, og generell x- og y-posisjon vil være hensiktsmessig. Dette kan vises med å legge på en ekstra dimensjon til visualiseringen.

Visualisering over tre plan gir oss muligheten til å se hvilken påvirkning blant annet den romlige spredningen har på klassifikasjonene. Den romlige spredningen måles på interaller over et sekund, med tilsvarende utregningen som for ID-T algoritmen.

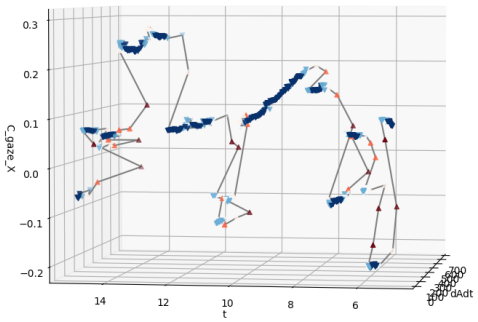
Figur 18: Taxi-Tour-Still visualisert med romlig spredning



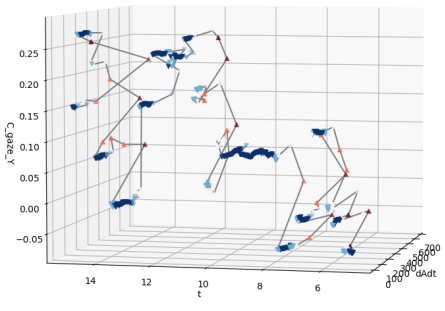
Visualisert med romlig spredning og vinkelhastighet

Figur 18 visualiserer den romlige spredningen sammen med den gjennomsnittlige klassifikasjonen til algoritmene. Fra figuren er det vanskelig å lese hvorvidt den romlige spredningen kan sies å ha direkte innflytelse på vinkelhastigheten og hvordan algoritmene overordnet klassifiserer øyedataen. Siden den romlige spredningen regnes ut med et sekunds intervall, ønsker vi å se på forskjellen av intervaller med høy romlig spredning, sammenlignet med lav romlig spredning. Fra figur 18 man det til en viss grad se ut som at høy spredning gir noe høyere vinkelhastighet, da spesielt i tidsrommet 10 til 15 sekunder. Likevel ser vi at intervall med lav romlig spredning mellom 5 og 8 sekunder, også klassifiserer en del sakkader. Dette er interessant nok tidsintervallet som ga mye usikkerhet mellom algoritmene, som presentert i figur 15. Det er ikke nok med kun visualisering for å avgjøre påvirkningen til romlig spredning.

Figur 19: Taxi-Tour-Still x-akse



Figur 20: Taxi-Tour-Still y-akse



Dersom man erstatter den romlige med blikkets posisjon langs x- og y-aksen, er det mulig å se tydeligere forskjeller. Romlig spredning er en beregning gjort basert på endring i blikket over tid, slik at figur 19 og figur 20 også blir en representasjon på hvordan endring i blikket påvirker klassifisering. En kan se at perioder med lite endringer i blikkets posisjon gir lav endring i vinkelhastighet, og at større endringer i posisjon gir høyere hastighet og flere sakakder. Det er viktig å nevne her at liten spredning i blikkets posisjon fortsatt kan gi raske øyebevegelser, da blikket kan bevege seg raskt innenfor et lite område. Figurene viser om ikke annet, at selve posisjonen til blikket potensielt kan inkluderes i videre analyse.

4.1.1.3 Dataanalyse og feature selection

Fra litteraturen og visualisering av egen data, vet vi nå at både vinkelhastighet og romlig spredning i form av blikkets posisjon er to svært sentrale mål som kan brukes for å skille forskjellige typer øyebevegelser. Fra utviklingsprosessen til de nye eksperimentene, samt dataen som er samlet inn og visualisert, er det også klart at det er forskjeller i adferden til øyet ved gjennomføringen av de enkelte scenene.

Likevel er ikke dette et godt nok utgangspunkt for å konkludere med hvilke parametere man må beholde, og hvilke som kan forkastet. Det er nødvendig å kjøre en dypere analyse, med spesielt to fokusområder:

1. Kan vi avgjøre hvilke data som er mest relevant for hvilke øyebevegelser som klassifiseres?
2. Kan vi se hvilke parametere som potensielt har størst innflytelse på hvilke scener vi ser på?

En feature selection ble gjort på parameterne som er lagret for scenene, etter klassifiseringen. Et utvalg av de mest relevante resultatene presenteres her;

Tabell 5: Feature analyse på scener

Feature		MBD	MBS	OBM	OBS
dAdt	mean	48.7	54.5	28.9	27.6
	std	96.3	102.9	67.8	69.7
dAangle	mean	0.57	0.61	0.36	0.30
	std	1.19	1.21	0.84	0.77
dVdt	mean	237	256	149	135
	std	9282	9775	6331	6362
dispersion	mean	10.6	13.1	8.56	9.90
	std	7.65	9.43	7.36	7.13
average	mean	1.13	1.16	1.08	1.07
	std	0.270	0.281	0.216	0.228
gaze X	mean	0.00956	0.00547	0.00444	0.0240
	std	0.111	0.125	0.126	0.108
gaze Y	mean	-0.0385	-0.0195	0.0253	0.0169
	std	0.0979	0.121	0.107	0.117

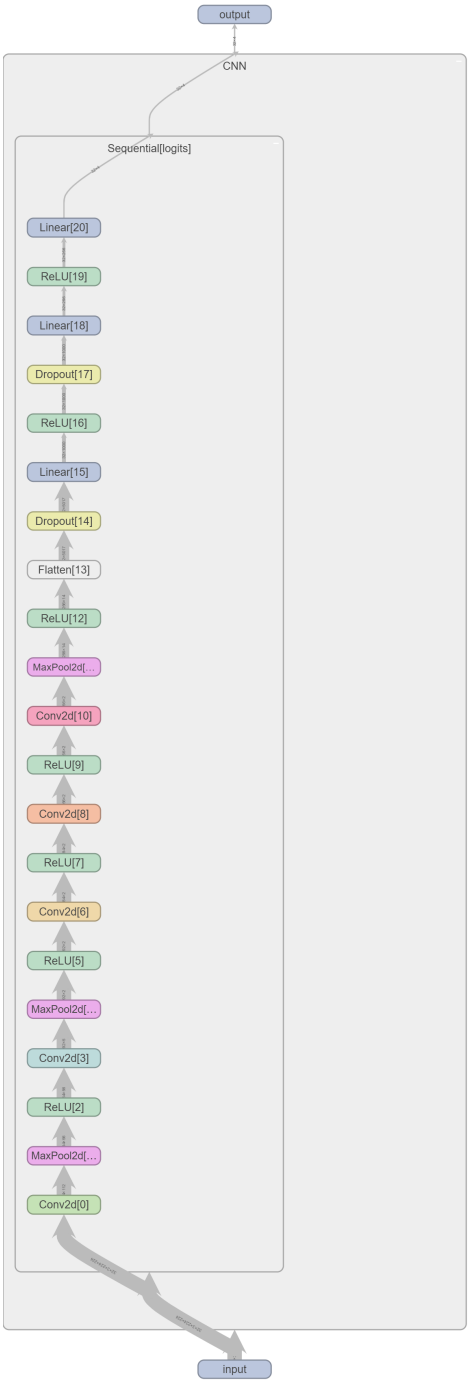
Gjennomsnittlig verdi for features med standardavvik for hver scene

Vinkel, vinkelhastighet og vinkelakselerasjon viser alle store utslag mellom eksperimentene med flere objekter, og eksperimentene med et enkelt objekt. Romlig spredning gir også noe forskjellige resultater mellom scenene, og er aller høyest for many-ball-still. Den gjennomsnittlige klassifiseringen brukes i analysen, og forteller oss at det som forventet er en tydelig korrelasjon mellom vinkelhastigheten til øyet, og til dels også den romlige spredningen. Scenene med flere objekter gir en høyere gjennomsnittlig vinkelhastighet, enn scenene med kun en ball. Det klassifiseres også flere sakkader for scener med flere objekter, og mest av alle for Many-Ball-Still. En gjennomsnittlig måling for blikkets posisjon i rommet er i seg selv ikke et egnet mål for å avgjøre hverken øyebevegelser eller scener. Om en inkluderer det temporale aspektet i analysen, er det vist at det også blikkets posisjon kan være et nyttig mål som likevel kan brukes.

4.1.2 Konvolusjonelt Nevralt Nettverk

4.1.2.1 Arkitektur

Gruppen endte med et konvolusjonelt nevralt nettverk som over flere iterasjoner ble utviklet for høyest grad av presisjon og pålitelighet. Mellom lagene benyttes max pooling for å redusere størrelsen til modellen. Dropout benyttes som regulariseringsmetode, slik at enkelte lærte hidden layer nevroner ignoreres under trening. Dette er ment for å forhindre overfitting av modellen.

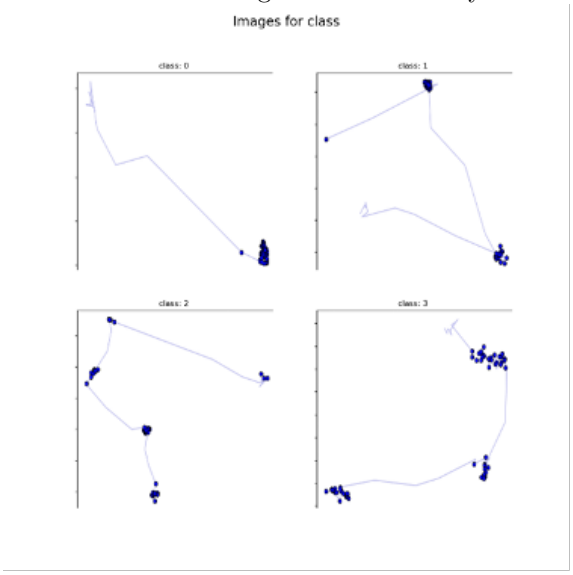


Figur 21: CNN arktitektur

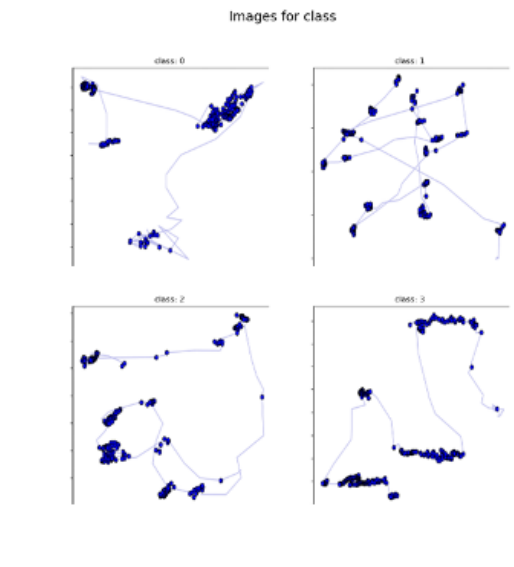
4.1.2.2 Visualisering

Etter datasettet er prosessert, er det mulig å gjøre en kvalitativ analyse for å skape en bedre forståelse av de kommende kvantitative resultatene fra maskinlæringsmodellen.

Figur 22: Bilder benyttet under trening av CNN-modellen



Bilder for 1 sekunders data



Bilder for 5 sekunders data

Ved å se på øyedataen er det et par trekk man kan se når man har en forståelse for hvordan de forskjellige scenene påvirker øyebevegelsene. One-ball-still ble laget for å få fikseringer i ro i et punkt av gangen, før blikket hurtig flyttet seg videre til neste punkt. Mant ball disappear vil ha blikket fiksert i ro i et område om gangen, men vil hurtig forflytte seg ettersom objektene nesten umiddelbart endret plass. Mange baller stille har flere punkter å fiksere på. En ball bevegelse har et og et punkt å fiksere på, som sakte beveger seg i rommet.

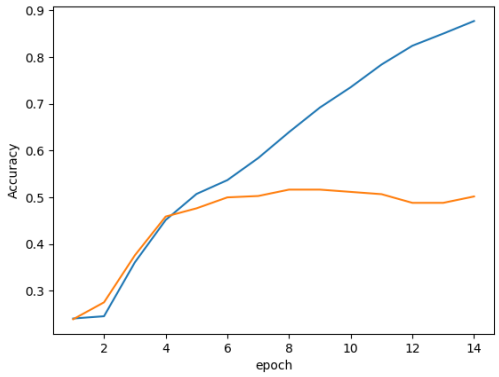
Figurene fra både et- og fem sekunders intervaller viser at det er mulig å kjenne igjen noen av disse egenskapene direkte fra figuren. Spesielt fra figurene over fem sekunder ser man hvor blikket har vært i ro, hvor det sakte har beveget seg i rommet, og hvor det raskt har endret posisjon.

4.1.2.3 Presisjon og tap

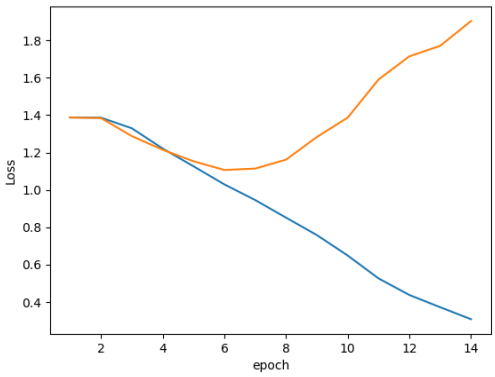
Treningen ble satt til å gå gjennom et maksimalt antall 20 epoker. For å spare unødvendig tid til trening ble det bestemt at modellen skulle stoppe å trene dersom valideringspresisjonen ikke så noen forbedring etter 5 påfølgende epoker. Gjennom kjøringene lagres de beste modellene automatisk, slik at det ikke var et behov for å vite hvor lenge en trengte å kjøre treningen før den begynte. Den beste modellen defineres i dette tilfellet som en modell med høyere presisjon, eller lavere loss for validering enn den tidligere beste modellen. Det er den beste modellen som brukes til slutt, for å bekrefte resultatene på det uavhengige testsettet.

Et sekunds øyedata

Et sekunds data består av 4965 bilder på totalt 132MB. Dataen er fordelt på 2979 bilder til trening, og 993 hver til validering og testing. Fordelingen av klasser til treningen er 766, 756, 738 og 719 for klasse 0 til 4. Den høyeste mulige presisjonen for å gjette en vilkårlig klasse er derfor 766 delt på 2979, som gir 0.2571.



Figur 23: Presisjon trening og validering 1 sekund



Figur 24: Tap trening og validering 1 sekund

Figur 23 og 24 viser resultatene fra kjøringen på modellen når trening-, validering-, og testdatasettet var satt til å være et bilder av øyedataen og tilhørende fikseringer over et sekund. Fra grafene for presisjon og loss kan man umiddelbart se at treningssettet får betraktelig bedre resultater enn valideringssettet. Etter om lag fem epoker fortsetter presisjon og loss for treningssettet å henholdsvis stige og synke. For valideringssettet på den andre siden, divergerer presisjonen på rundt 50 prosent, mens tap begynner å øke.

Tabell 6: Resultater 1 sekund

Epoke	Train acc	Valid acc	Train loss	Valid loss
1	0.240802	0.239258	1.387507	1.387201
2	0.245678	0.275391	1.386476	1.386476
3	0.360926	0.375977	1.329516	1.287904
4	0.451574	0.458984	1.221365	1.214738
5	0.507314	0.476562	1.126296	1.153352
6	0.536902	0.500000	1.029589	1.106585
7	0.584109	0.502930	0.945859	1.114179
8	0.639628	0.516602	0.851676	1.161847
9	0.692154	0.516602	0.758456	1.283102
10	0.735705	0.511719	0.649106	1.386663

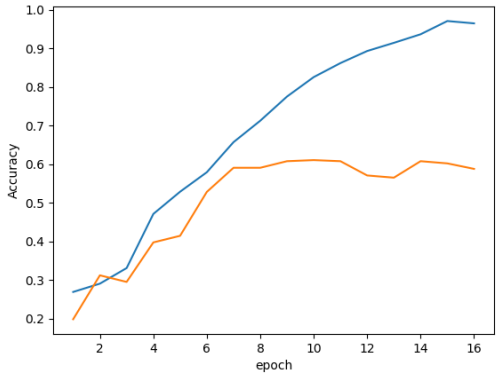
Presisjon og tap for trening- og valideringssett epoke 1-10 for 1 sekund

For å nærmere illustrere adferden i større detalj, kan vi se på tallene fra de første epokene, hvor presisjon og tap følger hverandre tett. Senere i treningen, ser vi skillet mellom de to blir større, og tap begynner å øke kraftig for trening. Den beste valideringspresisjonen med lavest loss får vi i epoke 8, med 51.66 prosent presisjon. Dette er også det punktet for tapsgrafen til valideringssettet snur, og begynner å øke.

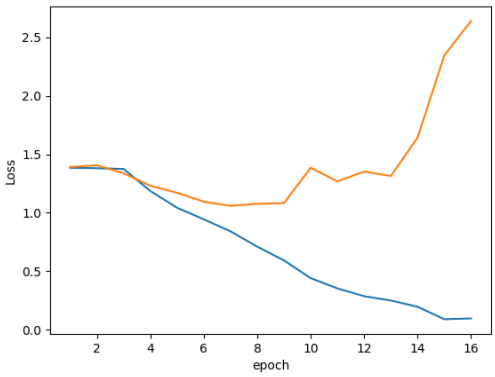
Ved å teste modellen som lagres etter epoke 8 på et uavhengig testsett får vi en presisjon på 47.75 som er noe lavere enn presisjonen på valideringssettet.

Tre sekunders øyedata

Ved å utvide intervallet til tre sekunder, får vi et mer detaljert bilde, men færre bilder å trene på. Totalt inneholder dette datasettet 1605 filer, som utgjør totalt 57mb. Dataen er fordelt med 963 bilder til trening, og 321 bilder hver til validering og testing. Fordelingen av klasser er 262, 212, 239 og 250, som gir en maks presisjon av å gjette en vilkårlig klasse på 262 / 963 = 0.2721.



Figur 25: Presisjon trening og validering 3 sekund



Figur 26: Tap trening og validering 3 sekund

Umiddelbart fra observasjonene, er det tydelig at resultatene er noe bedre for tre sekunder, sammenlignet med et sekunder. Presisjon for valideringssettet holder seg jevnt med treningsettet lengre enn tidligere, før presisjonen til slutt stagnerer omkring epoke 8. Tap for valideringssettet er fortsatt relativ høyt, med noe stigning som starter ved epoke 10. Den går likevel ikke over det initiale tapet på 1.4 før etter epoke 12, i motsetning til et sekunds hvor stigningen har vært tydeligere ved tidligere epoker.

Tabell 7: Resultater 3 sekunder

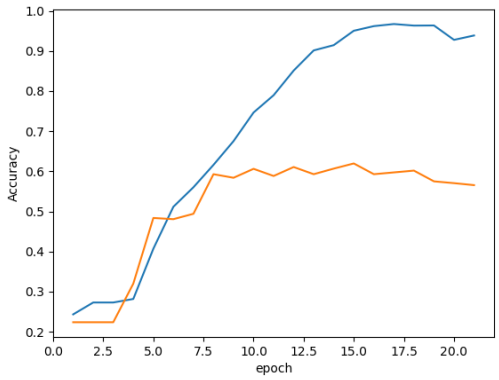
Epoke	Train acc	Valid acc	Train loss	Valid loss
1	0.269489	0.198864	1.385050	1.391890
2	0.290995	0.312500	1.381227	1.406171
3	0.331317	0.295455	1.374566	1.338136
4	0.471438	0.397727	1.186519	1.230906
5	0.528898	0.414773	1.042619	1.170998
6	0.579301	0.528409	0.943602	1.095030
7	0.657258	0.590909	0.840511	1.059058
8	0.712702	0.590909	0.709452	1.076525
9	0.775202	0.607955	0.591930	1.082627
10	0.825605	0.610795	0.439707	1.385383
11	0.861895	0.607955	0.352030	1.268525
12	0.893145	0.571023	0.285352	1.352151

Presisjon og tap for trening- og valideringssett epoke 1-12 for 3 sekund

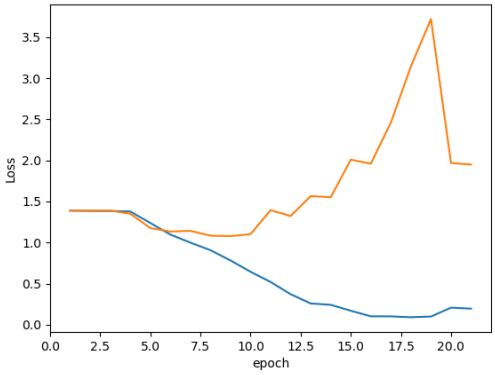
Resultatene fra tre sekunders intervallet er gode sammenlignet med tidligere resultater. Også i dette tilfellet kan det se ut som at modellen overfitter, da både presisjon og tap faller et godt stykke bak treningssettet etter hvert som den trener gjennom forskjellige epoker. Den høyeste presisjonen for valideringssettet kommer i epoke 10, hvor den klassifiserer riktig scene av de fire mulige 61.1 prosent av gangene. Sammenlignet med om lag 50 prosent fra tidligere, er dette et svært positivt resultat. Også på det uavhengige testsettet klassifiserer den riktig over 60 prosent av gangene, med en presisjon på 63.35 prosent.

Fem sekunders øyedata

Datasettet inneholder 1005 bilder på totalt 44MB. Disse er fordelt på 603 til trening og 201 til validering og testing. Antallet for de forskjellige klassene er: 141, 153, 165 og 144, som gir en maksimal presisjon på 165 delt på 603 er 27.36 prosent ved vilkårlig gjetting av kun klasse 2.



Figur 27: Presisjon trening og validering 5 sekund

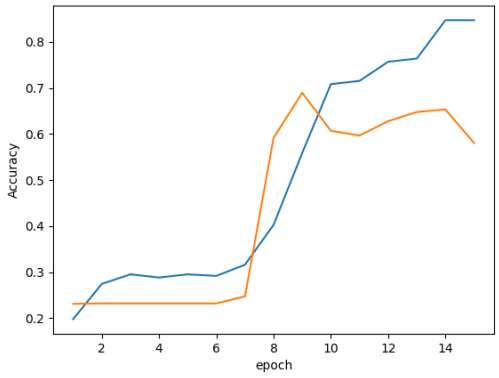


Figur 28: Tap trening og validering 5 sekund

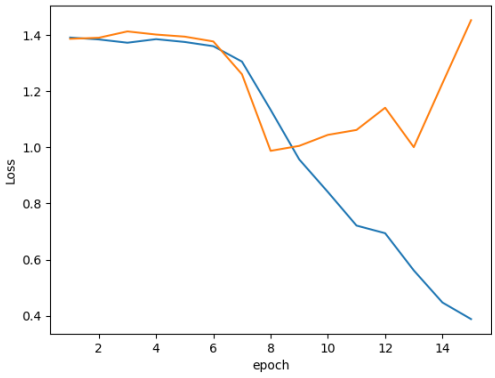
Resultatene fra 5 sekunders data er i stor grad sammenlignbare med resultatene fra 3 sekunders dataen. Som for tre sekunder stabiliserer presisjonen seg etter om lag åtte epoker. Den beste presisjonen kommer på epoke 15, og ligger på 0.6195. På det uavhengige testsettet gir det en presisjon på 62.70 prosent, som også er veldig likt tre sekunders modellen.

10 sekunds øyedata

Datasettet med bilder på 10 sekunder er på 429 bilder og utgjør 25MB. Det er satt av 258 bilder til trening, 86 til validering og 85 til trening. Fordelingen av treningsdataen er på 67, 64, 58 og 70 for de fire klassene, som gir minimumspresisjon for vilkårlig gjetting 27.13 prosent.



Figur 29: Presisjon trening og validering 10 sek



Figur 30: Tap trening og validering 10 sek

Fra figur 29 og 30 ser vi at presisjonen er svært lav før epoke 6, hvor den skyter opp i været mot epoke 8. Etter dette begynner den å gradvis synke, samtidig som tapet skyter oppover.

Tabell 8: Resultater 10 sekunder

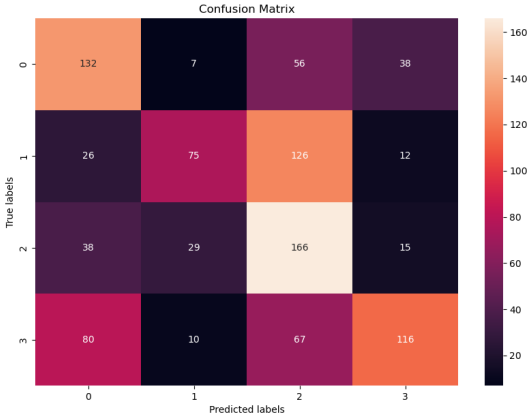
Epoke	Train acc	Valid acc	Train loss	Valid loss
6	0.291667	0.232008	1.360664	1.377454
7	0.315972	0.247159	1.305683	1.260563
8	0.402778	0.591856	1.133959	0.987751
9	0.559028	0.689394	0.957078	1.005412
10	0.708333	0.607008	0.841056	1.044643
11	0.715278	0.596591	0.721478	1.062121

Presisjon og tap for trening- og valideringssett epoke 6-11 for 10 sekund

Epoke 6 til 11 viser den interessante adferden til treningen. Valideringssettet presterer bedre enn treningssettet frem til epoke 10, før presisjon begynner å synke, mens loss øker. Beste presisjon ser vi i epoke 9, med nesten 70 prosent på valideringssettet. Dette ser derimot ut til å være et spesialtilfelle, tatt resultatene rundt til betraktning. Om vi prøver modellen på det uavhengige testsettet, får vi en presisjon på 47.57 prosent som er betraktelig svakere enn valideringspresisjonen, og dermed bekrefter mistanken om at modellen overfitter.

4.1.2.4 Forvirringsmatrise

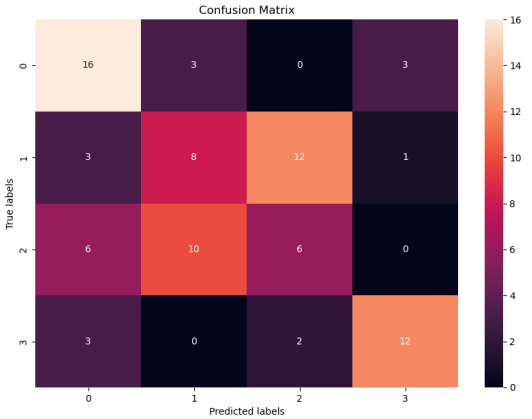
En forvirringsmatrise kan være et nyttig verktøy for å visualisere styrker og svakheter til en modell. Denne gir en oversikt over hvor en modell har klassifisert, og hvor ofte den klassifiserer feil mellom to spesifikke klasser. 1 sekund og 10 sekunder var de to datasettene som gjorde det dårligst:



Figur 31: Forvirringsmatrise 1 sekund

0: one ball still, 1: many ball disappear, 2: many ball still, 3: one ball move

Et sekunds modellen klassifiserer klasse 2 many-ball-still mer enn noen andre klasse. 415 av de totalt 993 klassifikasjonene var mot klasse 2, som er i overkant av 40 prosent. Den sliter med å predikere klasse 1 many-ball-disappear, som den kun tror er riktig 121 ganger.

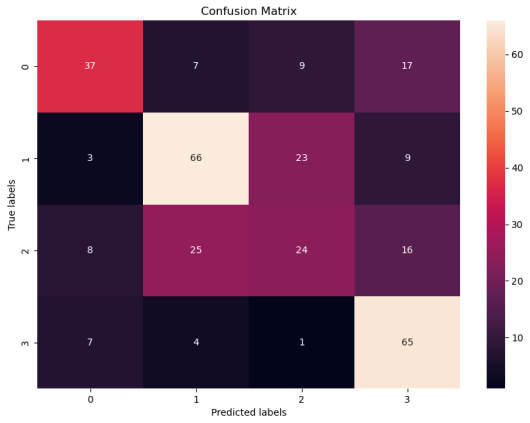


Figur 32: Forvirringsmatrise 10 sekund

0: one ball still, 1: many ball disappear, 2: many ball still, 3: one ball move

Ti sekunder modellen har betraktelig færre klassifiseringer som følge av det begrensede antallet

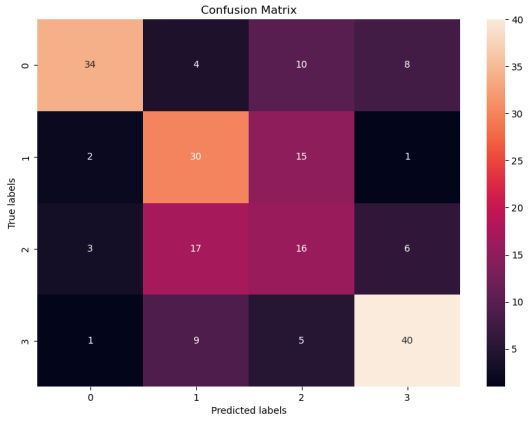
med bilder i datasettet. Denne modellen hadde også svært stor forskjell mellom presisjonen på valideringssettet, og klassifisering på testsettet, og gikk ned fra 68.9 til 47.5 prosent riktige klassifiseringer. Modellen klassifiserer klasse 0 og 1 riktig mesteparten av gangene, mens den sliter med å predikere riktig mellom klasse 2 og 3. Her er det faktisk slik at de oftere klassifiserer klassen som ikke er riktig. De to modellene som gjorde det desidert best var modellene på tre- og fem sekunders datasettet. Disse hadde svært lik presisjon på beste epoke for både test- og validering. Også utvikling over epoker var svært sammenlignbar mellom disse, både for presisjon og loss.



Figur 33: Forvirringsmatrise 3 sekund

0: one ball still, 1: many ball disappear, 2: many ball still, 3: one ball move

Tre sekunders forvirringsmatrise viser oss et betraktelig bedre sett med klassifiseringer for modellen sammenlignet med 1- og 10 sekunders modellene. Klasse 1 og 3 blir predikert riktig store deler av gangene. Også klasse 0 blir predikert riktig store deler av gangene. Denne modellen, i likhet med ti sekunders, verger seg fra å gjette klasse 2, og klassifiserer heller ikke spesielt nøyaktig når den først gjør det.



Figur 34: Forvirringsmatrise 5 sekund

0: one ball still, 1: many ball disappear, 2: many ball still, 3: one ball move

Fem sekunders forvirringsmatrise gir veldig like resultater som for tre sekunder. Klasse 0, 1 og 3 predikeres riktig mesteparten av tiden, mens klasse 2 i mye større grad predikeres vilkårlig.

Fra resultatene presentert i forvirringsmatrisene er det åpenbart at de to klassene CNN-modellene sliter mest med å skille på, er many-ball-disappear og many-ball-still. Disse er de eneste to scenene

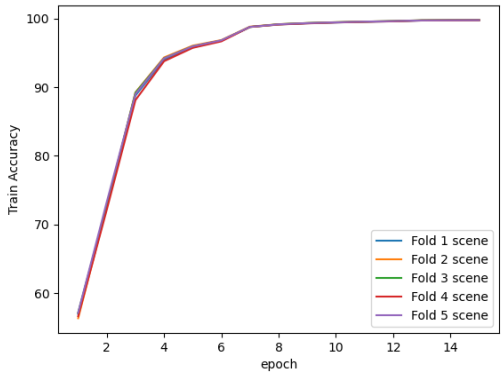
med flere enn et potensielt fikseringspunkt.

4.1.3 Reccurent Nevrale Nettverk

For RNN klassifisering ble det benyttet en Long-Short-Term Memory modell fra PyTorch biblioteket. Datasettet ble delt inn i 80 prosent trening og 20 til validering. Totalt var dette om lag 30 minutter med øyedata per scene, dedikert til trening og validering. Det var også samlet inn data som var helt uavhengig fra datasettet som ble trent på. Denne dataen tilsvarte om lag 10 prosent av det totale datasettet som ble benyttet under trening og validering. I dette delkapittelet vil det bli presentert resultater fra kjøringer på klassifisering av scener samt testpersoner, og hvordan enkelte endringer som ga høyere presisjon og større grad av brukbarhet for de endelige modellene.

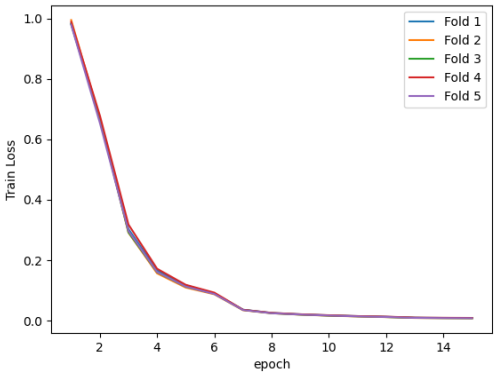
4.1.3.1 Klassifikasjon av scener

Figur 35: Presisjon trening



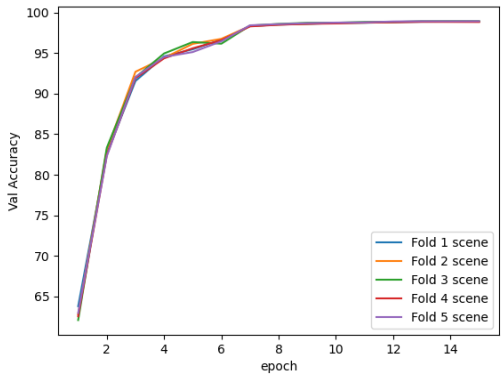
Presisjon på trening av scener med fem folds

Figur 36: Tap trening



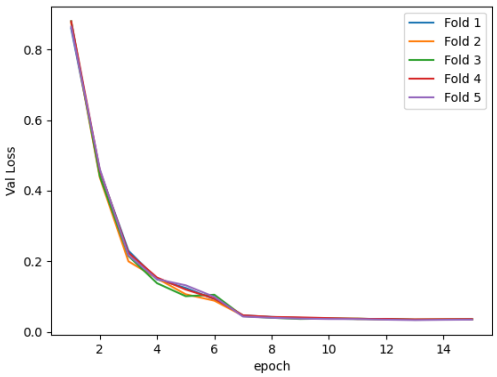
Tap på trening av scener med fem folds

Figur 37: Presisjon validering



Presisjon på validering av scener med fem folds

Figur 38: Tap validering



Presisjon på validering av scener med fem folds

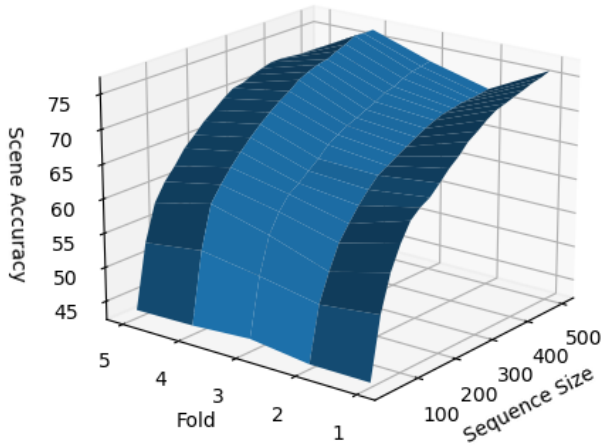
Tabell 9: Resultater Klassifikasjon av scener

Epoke	Train acc	Valid acc	Train loss	Valid loss
1	0.572	0.629	0.979	0.864
2	0.733	0.823	0.655	0.461
3	0.890	0.921	0.296	0.217
4	0.942	0.945	0.161	0.149
5	0.959	0.951	0.112	0.132
6	0.968	0.965	0.088	0.099
7	0.988	0.984	0.035	0.043
8	0.991	0.986	0.025	0.040
9	0.993	0.988	0.020	0.037
10	0.994	0.988	0.017	0.036
11	0.995	0.988	0.014	0.036
12	0.996	0.989	0.012	0.034
13	0.997	0.989	0.009	0.033
14	0.997	0.989	0.009	0.034
15	0.997	0.990	0.008	0.035

Presisjon og tap for trening- og valideringssett epoke 1-15 for klassifikasjon av scener på fold 5

Figurene viser at modellen gir svært gode resultater for både trening og validering. Foldene presterer også ganske likt, slik at en ikke umiddelbart trenger å mistenke at resultatene ikke er reelle. Trening og validering ender både på 99 prosent, mens tap for begge synker i takt med epokene, dog noe lavere treningssettet. Som nevnt i metodedelen er det slik at trening og validering deles opp fra samme datasett som på forhånd er blitt stokket. Det er derfor viktig å være varsom om man skal dra konklusjoner bare basert på valideringssettet, da den potensielt kan ha trent slik at den ikke er godt egnet for å ta inn ny usett data. Det er derfor hensiktsmessig å se på presisjon per fold på det uavhengige testsettet.

Figur 39: Sekvenslengde for forskjellige folds



Presisjon på testsettet for forskjellig sekvenslengde per fold

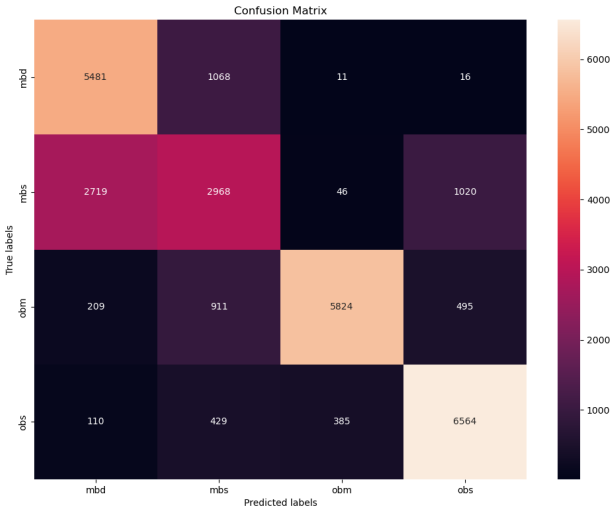
Tabell 10: Resultater fra uavhengig testsett

Fold	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
beste sekvenslengde	500	500	500	450	400
Presisjon	75.998	71.489	73.743	76.684	72.470
Gjennomsnitt	74.077				
Standardavvik	1.994				

Presisjon på uavhengig testsett for forskjellige fold, med gjennomsnitt og standardavvik

Fold fire gjør det best med en presisjon på 76.68 prosent. Den gjennomsnittlige presisjonen på tvers av foldene er 74 prosent. Den gir et standardavvik på 1.99, som tyder på at resultatene fra testsettet er til å stole på. Likevel er det åpenbart resultatene på det uavhengige at modellen fungerer dårligere når den får usett data. 76 prosent er svært mye høyere enn vilkårlig gjetting, og er fortsatt betraktelig bedre enn de beste resultatene fra CNN-modellen. Det er derfor viktig å analysere videre for å se hvor feilene oppstår.

Figur 40: Forvirringsmatrise Scener



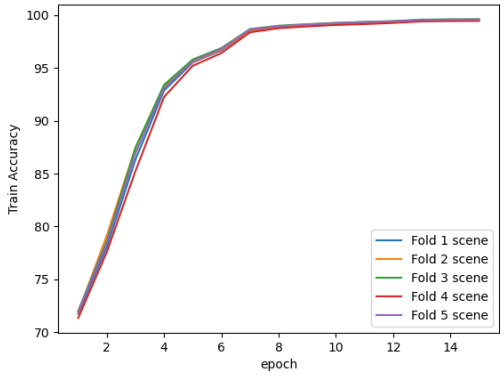
Forvirringsmatrise for fire scener

Forvirringsmatrisen 40 til testingen viser noen av de samme tendensene som CNN-modellen. One-ball-move og one-ball-still klassifiseres riktig mest. Den har fortsatt vansker med å skille Many-ball-disappear og Many-ball-still. Det skal sies at den oftere klassifiserer rett enn galt, men det blir tydelig at disse datasettene er svært vanskelig å skille. De to scenene er relativt like, da begge ikke nødvendigvis har et spesifikt mønster testpersonene skal følge. Vi har også sett på dataen som visualiseres at det kan være vanskelig å skille disse også for mennesker. Gruppen ønsker derfor å se hvordan modellen med tre scener som en med stor grad av sikkerhet kan si gir ulik type øyedata.

4.1.3.2 Klassifikasjon av distinkte scener

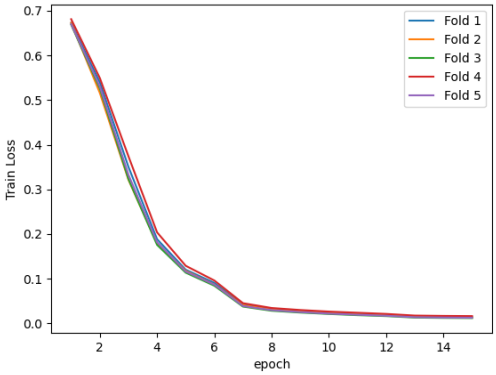
Ved å eliminere many-ball-still, sitter man igjen med tre scener som har tre ulike og konkrete hensikter. One-ball-still skal ha fiksering i et lite område over tid, One-ball-move vil ha jevn forflytning over tid, Many-ball-disappear vil ha ballistiske sakkader slik at en testperson ikke kan hvile blikket over tid. Resultatene som følger, vil vise hvordan modellen presterer gitt dette nye utgangspunktet.

Figur 41: Presisjon trening



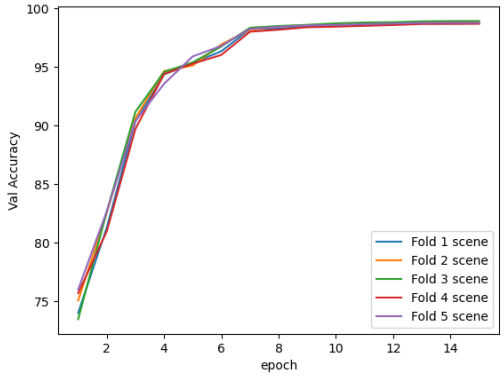
Presisjon på trening av scener med fem folds

Figur 42: Tap trening



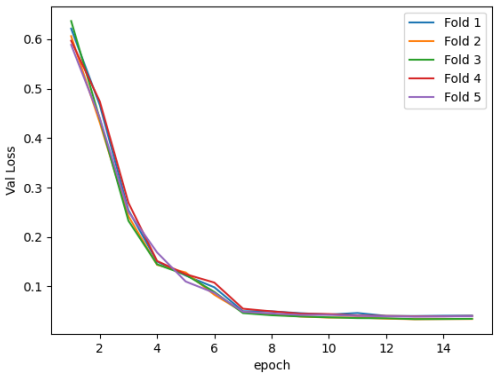
Tap på trening av scener med fem folds

Figur 43: Presisjon validering



Presisjon på validering av scener med fem folds

Figur 44: Tap validering



Presisjon på validering av scener med fem folds

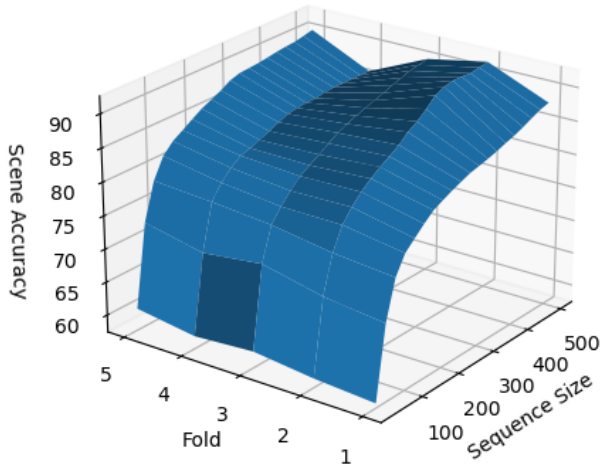
Tabell 11: Resultater Klassifikasjon av distinkte scener

Epoke	Train acc	Valid acc	Train loss	Valid loss
1	0.719	0.760	0.670	0.588
2	0.784	0.827	0.530	0.440
3	0.869	0.904	0.333	0.250
4	0.931	0.936	0.182	0.167
5	0.956	0.959	0.116	0.110
6	0.968	0.968	0.087	0.087
7	0.986	0.982	0.039	0.049
8	0.989	0.984	0.030	0.045
9	0.991	0.985	0.025	0.042
10	0.992	0.986	0.022	0.041
11	0.993	0.986	0.019	0.040
12	0.994	0.987	0.017	0.040
13	0.995	0.988	0.014	0.039
14	0.996	0.988	0.013	0.039
15	0.996	0.988	0.013	0.040

Presisjon og tap for trening- og valideringssett epoke 1-15 for klassifikasjon av tre scener på fold 5

Interessant nok er det ingen merkbar forskjell i verken presisjon eller tap for hverken trenings eller valideringssettet. Det skal merkes at presisjonen allerede var svært høy, slik at det naturligvis vil være vanskelig å øke presisjon ytterligere fra over 99 prosent. Fra forrige delkapittel vet vi at forbedringen må komme på det uavhengige testsettet for at modellen skal kunne bli enda mer anvendbar.

Figur 45: Sekvenslengde for forskjellige folds



Presisjon på testsettet for forskjellig sekvenslengde per fold

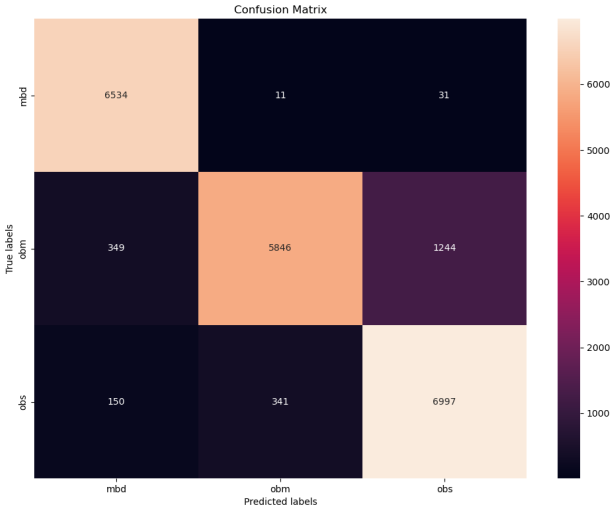
Tabell 12: Resultater fra uavhengig testsett

Fold	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
beste sekvenslengde	500	500	500	400	500
Presisjon	88.016	91.862	90.113	96.128	90.262
Gjennomsnitt	89.276				
Standardavvik	1.993				

Presisjon på uavhengig testsett for forskjellige fold, med gjennomsnitt og standardavvik

Ved å se på resultatene fra testsettet i tabell 12, ser en at modellen presterer svært godt for å skille mellom de tre scenene, med over 90 prosent presisjon for fire av fem folds. Fold 4 gir absolutt høyeste resultat, med presisjon på over 96 prosent. Gjennomsnittspresisjonen ligger rett i underkant av 90, og med et standardavvik på 1.993 så tyder det på at akkurat fold fire kan gi et resultat noe over hva en kan forvente. Likevel er over 90 prosent presisjon et svært godt resultat, med en tydelig forbedring fra 75 prosent der alle scenene var med i trening og testing. Det skal nevnes at modellen nå kun trenger å skille mellom tre forskjellige muligheter, i stedet for fire som tidligere. Det er derfor hensiktsmessig å se nærmere på hvor modellen gjør feilene sine.

Figur 46: Forvirringsmatrise



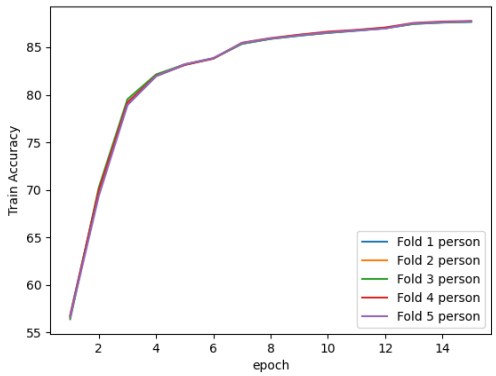
Forvirringsmatrise for tre scener

Forvirringsmatrisen viser at modellen i de aller fleste tilfeller nå klassifiserer many-ball-disappear riktig. Det er fortsatt noen ulikheter mellom one-ball-still og one-ball-move, som for så vidt er å forvente da disse deler en del trekk. Det er noen flere feil på tvers av disse, da enkelte tilfeller av feilaktig klassifisering av many-ball-still for one-ball-move har blitt til one-ball-still. Hvorvidt grunnlaget for å eliminere many-ball-still i klassifikasjonen er hensiktsmessig er for tidlig å konkludere med, og et viktig spørsmål å stille. Resultatene tyder på at modellen er verdt å utforske videre, både for å fortsette vurderingen av brukbarhet, samt se på forskjellige aktuelle anvendelser.

4.1.3.3 Klassifikasjon av testpersoner

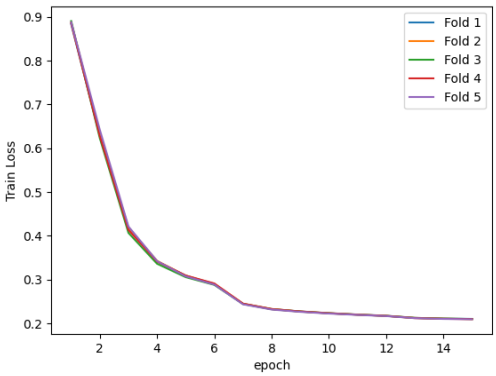
Det neste steget for å avgjøre brukbarheten til dataen i maskinlæring, blir å se på hvordan den presterer ved å skille på brukeren av HMD-et. Her vil modellen være avhengig av å kunne skille på de ulike tendensene til brukeren, i motsetning til å se på det spesifikke designet til scenene. I likhet med tidligere blir datasettet trent opp på de tre forskjellige gruppemedlemmene. De tre medlemmene har gjennomført de samme scenene, med de samme seedene over like lang tid.

Figur 47: Presisjon trening



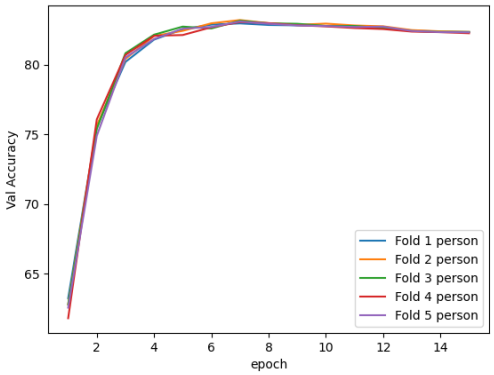
Presisjon på trening av testpersoner med fem folds

Figur 48: Tap trening



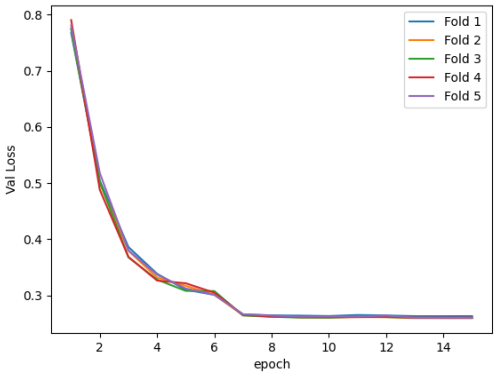
Tap på trening av testpersoner med fem folds

Figur 49: Presisjon validering



Presisjon på validering av testpersoner med fem folds

Figur 50: Tap validering



Presisjon på validering av testpersoner med fem folds

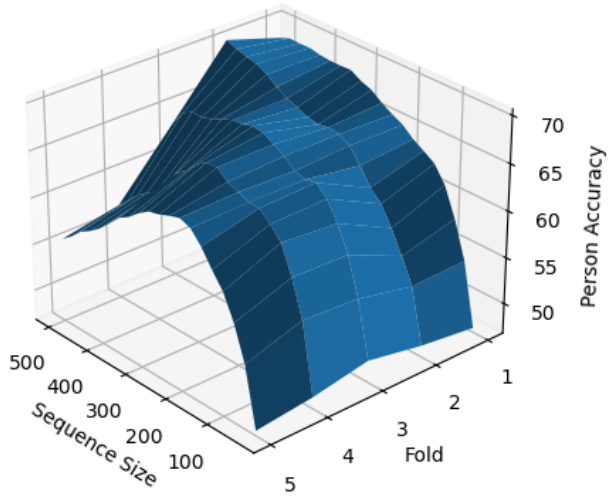
Tabell 13: Resultater Klassifikasjon av testpersoner

Epoke	Train acc	Valid acc	Train loss	Valid loss
1	0.566	0.626	0.888	0.780
2	0.694	0.749	0.642	0.518
3	0.789	0.805	0.421	0.380
4	0.820	0.818	0.342	0.337
5	0.832	0.826	0.307	0.312
6	0.839	0.827	0.288	0.301
7	0.854	0.831	0.243	0.266
8	0.859	0.829	0.231	0.264
9	0.863	0.828	0.226	0.262
10	0.865	0.828	0.222	0.262
11	0.868	0.827	0.219	0.264
12	0.870	0.827	0.216	0.264
13	0.875	0.824	0.211	0.260
14	0.877	0.823	0.210	0.260
15	0.877	0.823	0.209	0.260

Presisjon og tap for trening- og valideringssett epoke 1-15 for klassifikasjon av testpersoner på fold 5

Resultatene i tabell 13 er noe lavere enn tidligere, men gir likevel relativt høy presisjon selv for valideringssettet. Det kan se ut som at modellen overfitter litt etter epoke 6, men presisjonen holder seg jevn på rundt 82 prosent. Nok en gang er det viktig å se på de faktiske testresultatene.

Figur 51: Sekvenslengde for forskjellige folds



Presisjon på testsettet for forskjellig sekvenslengde per fold

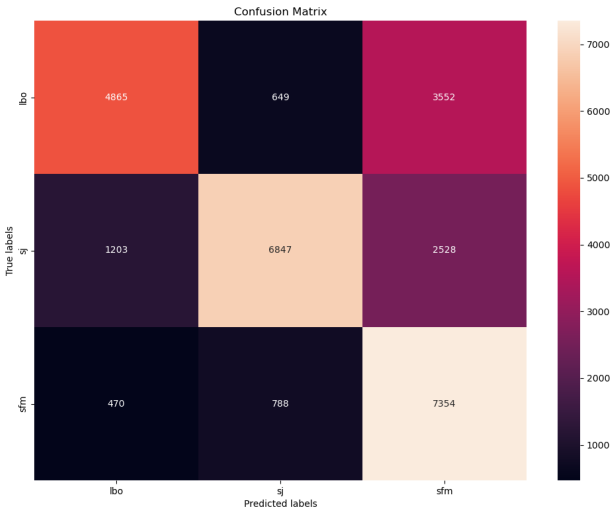
Tabell 14: Resultater fra uavhengig testsett

Fold	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
beste sekvenslengde	475	500	300	250	200
Presisjon	68.760	70.190	67.476	66.591	64.967
Gjennomsnitt	67.597				
Standardavvik	1.790				

Presisjon på uavhengig testsett for forskjellige fold, med gjennomsnitt og standardavvik

Presisjonen ligger et stykke under valideringssettet. På sitt beste presterer modellen 70 prosent på det uavhengige testsettet. Dette er nok en gang betraktelig bedre enn vilkårlig gjetting, og er isolert sett et ganske så godt resultat. Likevel er det ønskelig å se hvor modellen klassifiserer feil, og å utforske hvordan en eventuelt kan anvende modellen slik at en med høyere grad av sikkerhet kan skille på de ulike testpersonene.

Figur 52: Forvirringsmatrise



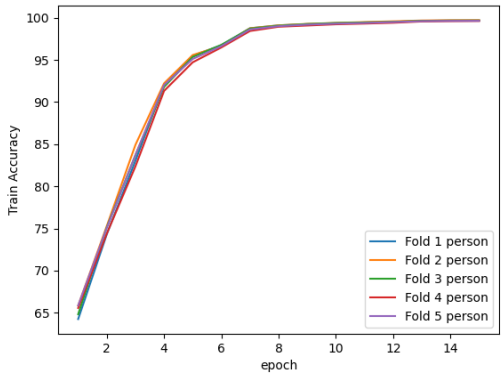
Forvirringsmatrise for testpersoner

Figuren lar oss se nærmere på hvor feilene oppstår. Det kan se ut til at datasettene fra lbo og sj er vanskelig å skille for modellen, mens sfm som regel blir klassifisert riktig. Når sant skal sies klassifiserer modellen riktig i de fleste tilfeller for alle testdeltagerne, som tyder på at det er et potensiale for å se forskjellen på øyebevegelsen på deltagerne også med høyere presisjon.

4.1.3.4 Klassifikasjon av testpersoner for spesifikk scene

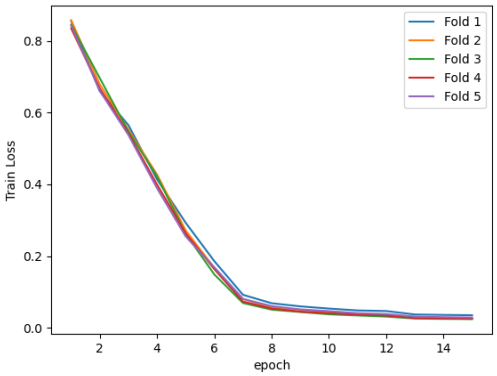
Et problem med å prøve å skille mellom testpersoner med en modell trent på mange forskjellige scener, er at den potensielt kan trene seg opp til å se forskjeller på de spesifikke scenen, i steder for testpersonene. Dette vil gi støy i dataen som kan bidra til at presisjonen blir lavere enn optimal. For å utforske om det er mulig å få høyere presisjon ble modellen også trent på en og en scene, slik at selve forskjellene på brukeren av HMD-et ble isolert.

Figur 53: Presisjon trening



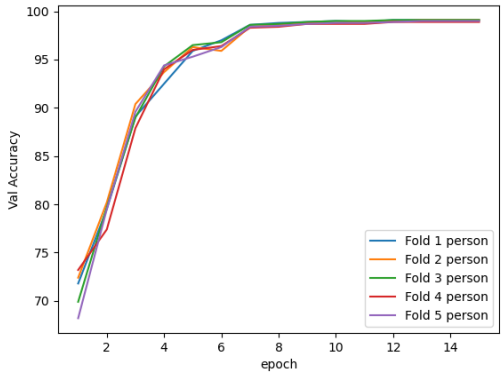
Presisjon på trening av testpersoner med fem folds

Figur 54: Tap trening



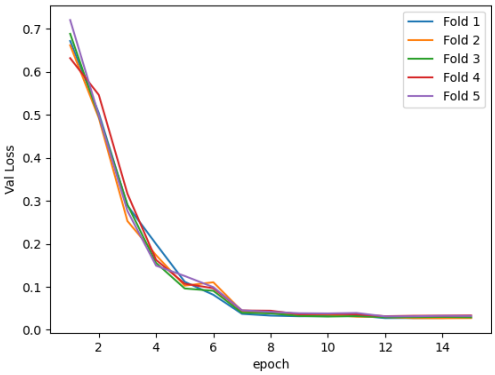
Tap på trening av testpersoner med fem folds

Figur 55: Presisjon validering



Presisjon på validering av testpersoner med fem folds

Figur 56: Tap validering



Presisjon på validering av testpersoner med fem folds

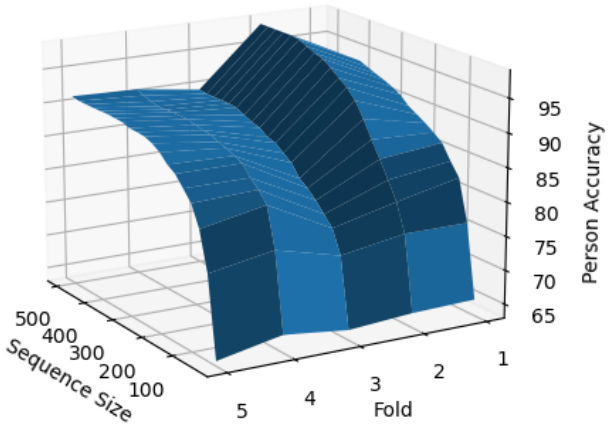
Tabell 15: Resultater Klassifikasjon av testpersoner

Epoke	Train acc	Valid acc	Train loss	Valid loss
1	0.659	0.682	0.767	0.720
2	0.751	0.795	0.591	0.498
3	0.837	0.896	0.407	0.276
4	0.920	0.944	0.212	0.149
5	0.951	0.953	0.132	0.125
6	0.966	0.963	0.092	0.099
7	0.986	0.984	0.039	0.045
8	0.990	0.985	0.029	0.042
9	0.992	0.987	0.023	0.038
10	0.993	0.988	0.020	0.039
11	0.994	0.988	0.018	0.031
12	0.995	0.989	0.015	0.031
13	0.996	0.990	0.012	0.032
14	0.996	0.990	0.011	0.032
15	0.996	0.990	0.011	0.032

Presisjon og tap for trening- og valideringssett epoke 1-15 for klassifikasjon av testpersoner i mbd på fold 5

Med denne endringen ble presisjonene på trening og validering betraktelig bedre. Der det tidligere ble en maksimal valideringspresisjon på rundt 82 prosent, med overfitting etter epoke 6, er det nå presisjon på 99 prosent med jevn økning utover kjøringen. Loss synker helt ned til 0.01 for trening og 0.03 for validering, og holder seg jevnt.

Figur 57: Sekvenslengde for forskjellige folds



Presisjon på testsettet for forskjellig sekvenslengde per fold

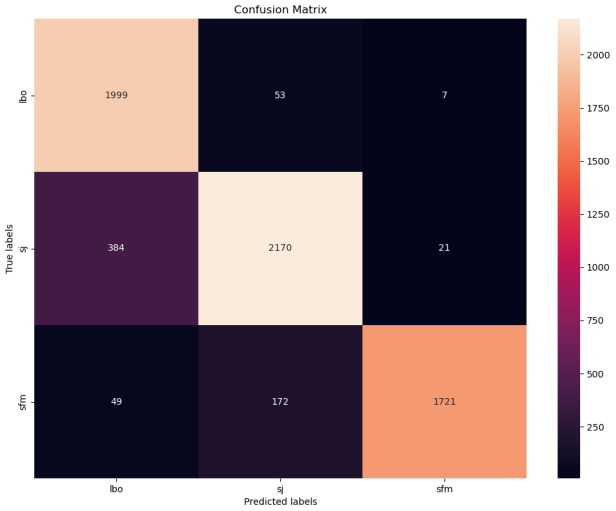
Tabell 16: Resultater fra uavhengig testsett

Fold	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
beste sekvenslengde	375	400	500	500	500
Presisjon	93.704	98.312	89.568	90.937	91.24
Gjennomsnitt	92.752				
Standardavvik	3.083				

Presisjon på uavhengig testsett for forskjellige fold, med gjennomsnitt og standardavvik

Selv på det uavhengige testsettet er resultatene nesten konsekvent over 90 prosent for alle folds, og har et gjennomsnitt på 93, med tre i standardavvik. Her ser vi tydelig at modellen tjente svært godt på å begrense seg til en og en scene, og many-ball-disappear ga spesielt gode resultater.

Figur 58: Forvirringsmatrise



Forvirringsmatrise for testpersoner én scene

Selv om det fortsatt er noen feilklassifiseringer hvor modellen gjetter lbo, der svaret sj, er modellen svært presis for alle tre testdeltagere, vist i figur 58.

4.1.3.5 Implementasjon i Unity

Å kjøre modellen i sanntid var ønskelig for videre bekreftelse på at resultatene fra treningen og testingen var reelle. Den beste presisjonen for de respektive modellene var sekvenslengde på mellom 350 og 500 datapunkter. Det ble derfor valgt å returnere klassifiseringer hvert femte sekund, scenene ble kjørt i et minutt for hver test. Modellene som ble brukt for å klassifisere var den beste modellen for de forskjellige bruksområdene. Det vil si klassifiseringsmodellen mellom one-ball-still, one-ball-move og many-ball-disappear for scener, og modellen for testpersoner trent på én scene many-ball-disappear. Scenene blir kjørt i 120 sekunder, og klassifiseres hvert femte sekund. Det blir klassifisert 22 ganger hver scene, da scenen avsluttet før modellen klassifiserer de siste fem sekundene.

Klassifisering av scener i unity

Tabell 17: Klassifisering av one-ball-still Unity

Scene	OBS	OBM	MBD
Antall gjett	18	3	1
Riktig	0.818		

Antall klassifiseringer av scener i Unity , One-Ball-Still

Tabell 18: Klassifisering av one-ball-move Unity

Scene	OBS	OBM	MBD
Antall gjett	10	11	1
Riktig		0.500	

Antall klassifiseringer av scener i Unity , One-Ball-Move

Tabell 19: Klassifisering av many-ball-disappear Unity

Scene	OBS	OBM	MBD
Antall gjett	0	2	20
Riktig			0.901

Antall klassifiseringer av scener i Unity , Many-Ball-Disappear

Klassifisering av testpersoner i unity

Tabell 20: Klassifisering av lbo Unity

Testperson	LBO	SJ	SFM
Antall gjett	16	4	2
Riktig	0.727		

Antall klassifiseringer av testperson i Unity , LBO

Tabell 21: Klassifisering av sj Unity

Testperson	LBO	SJ	SFM
Antall gjett	4	18	0
Riktig		0.818	

Antall klassifiseringer av testperson i Unity , SJ

Tabell 22: Klassifisering av sfm Unity

Testperson	LBO	SJ	SFM
Antall gjett	3	2	17
Riktig			0.773

Antall klassifiseringer av testperson i Unity , SFM

Som følge av at antall klassifikasjoner er svært få, kan en ikke bruke resultatene som en fasit på hvordan modellen håndterer klassifiserer i sanntid. Den gir derimot en indikasjon på hva den håndterer godt, og hva som er mer vanskelig. Først og fremst er det godt å se at den med veldig høy presisjon klassifiserer både one-ball-still og many-ball-disappear, dette gjør den nesten uten feil. One-ball-move er betraktelig vanskeligere i dette tilfellet, denne utfordringen kjenner vi igjen fra forvirringsmatrisen til den samme modellen. Klassifisering av testpersoner gjør det også overordnet svært bra. På scenen many-ball-disappear klassifiserer den som oftest riktig for alle tre testdeltagerne. Her er det ingen nevneverdig forskjell på de tre testpersonene, basert på det begrensede antallet klassifiseringer.

Utfordringer med implementasjon av modell i Unity

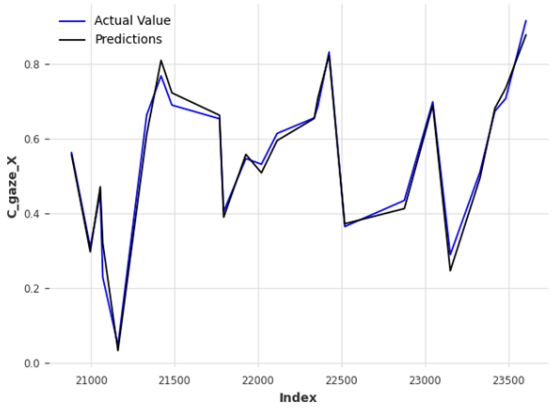
ONNX modellene som eksporteres og importeres i Unity er ikke spesielt stor, og burde heller ikke være spesielt krevende. Størrelsen på modellen ligger på 4.2MB, dette er ikke for stort for Unity å håndtere. Utfordringen ligger i hvordan klassifiseringen og oppdateringen av scenene håndteres. Unity sin innebygde onGetData funksjon henter informasjon for hver frame. Det blir kalt hver gang det skjer innhenting av data, samt for håndtering av objektene på scenen. Det er også denne som håndterer klassifikasjonene til maskinlæringsmodellene. Dette betyr at onGetUpdate må vente til klassifikasjonen er gjort, før resten av scenen kan begynne å oppdateres igjen. Det er ikke alltid dette er et stort problem, eller at testpersonen registrerer at scenen ikke er like responsiv som uten modellen. Derimot blir dette svært tydelig utover i eksperimentet, hvor ytelsen, spesielt for many-ball-disappear merkbart faller. Siden modellen klassifiserer basert på øyebevegelser, som i stor grad avhenger av den forhåndsbestemte adferden til scenene, vil dette naturligvis påvirke presisjonen. Det ble forsøkt implementert en måte å asynkront håndtere klassifiseringen, utenom de innebygde Unity-funksjonene, men da dette ikke fungerte etter en arbeidsdag ble det satt til side. Dersom arbeidet skal bygges videre på, er dette et konkret område som kan forbedres.

4.1.4 Time Series Forecasting

For prediksjon av sakkader ble biblioteket Darts benyttet, og en N-BEATS (Neural basis expansion analysis for interpretable time series) modell fra biblioteket ble implementert. Datasettet benyttet i trening av modellen ble delt opp i to, med 80 prosent av datasettet til trening og 20 prosent til validering. Settet med testsett besto av 93 sakkader som ble det opp i egne datasett som nevnt i metodekapittelet under punkt 3.2.7.2. De fleste sakkdene lå på mellom 10 til 20 datapunkter per datasett. Valideringssettene på sin side bestod av 24 lignende sakkader. Preprosessering av datasettet bestod av å benytte HMM algoritmen tidligere beskrevet i rapporten for å klassifisere et datasett. Sakkadene ble delt opp i egne datasett som nevnt for å muliggjøre multi-time series trening av modellen. Testing av modellen ble først utført på et separat datasett, for å deretter bli testet på alle datasett fra alle scener som gruppen hadde samlet inn over løpet av prosjektet. I dette delkapittelet vil resultatene fra disse testene bli presentert.

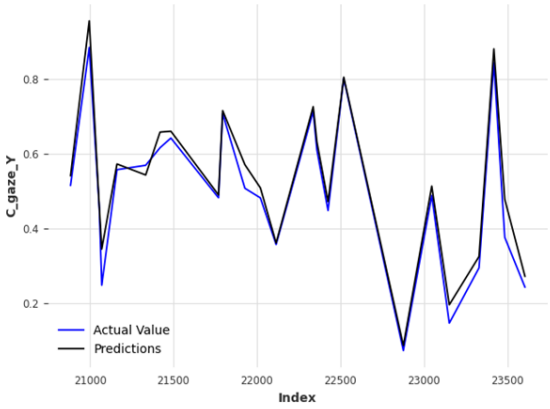
4.1.4.1 Forecasting av valideringssett

Figur 59: `C_Gace_X` vektor for validering



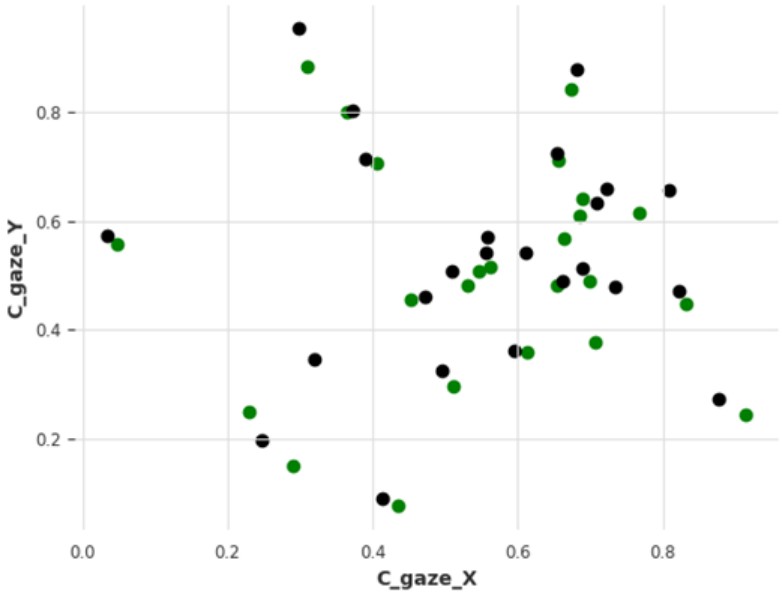
Forecasted- og den sanne verdi for blikkets x-vektor på valideringssettet

Figur 60: `C_Gace_Y` vektor for validering



Forecasted- og faktisk verdi for blikkets y-vektor på valideringssettet

Figur 61: Scatter plott for validering



Scatter: Viser forholdet mellom X- og Y-vektoren til blikket,

Figur 59 og Figur 60 viser en grafisk presentasjon av prediksjonen utført av modellen overlagt de faktiske verdiene funnet i datasettet. Spesifikt viser Figur 59 X-vektoren til testpersonens blikk, og tilsvarende viser Figur 60 Y-vektoren. Disse grafene viser resultatet fra prediksjon på valideringssettet. Med et endelig loss på 0.000762 skal det tilsi at modellen klassifiserer svært bra. Det samme blir reflektert i grafene hvor man ser at prediksjonen av en sakkades neste datapunkt ligger svært tett opptil hva de faktiske verdiene ser. X-asken er indeksen til sakkaden i det ordinalet datasettet, og representerer den temporale dimensjonen i datasettet. Figur 61 er et scatter plot som med X- og Y-vektor for blikket som de respektive aksene til plottet. Plottet viser forholdet mellom X- og Y-vektoren til blikket, og hvor i synsfeltet til brukeren sakkaden tok sted. Det skal bemerkes her at plottet dekker alle sakkader predikert, og man ser derfor ingen enkelt mønster i dataen, men heller individuelle gruppering av punkter som følger et mønster. Noen av grupperingen er mindre enn andre, enkelte bare et punkt, ettersom en sakkade ble satt til å måtte ha et minimum på 10 påfølgende punkter i datasettet for å bli tatt i bruk av modellen. Dette ble satt for å sørge for at det var nok punkter i hvert enkelt datasett for å gi datasettene nok punkter av verdi. For valideringssettet fikk prediksjonene en root mean squared error verdi på 0.032 på prediksjon av blikkets X-, Y- og Z-vektor. Tilsvarende fikk prediksjonen en gjennomsnittlig mean absolute percentage error på 0.07%. Disse verdiene kan virke svært lave, så det vil være naturlig å utføre de samme testene på et uavhengig datasett.

4.1.4.2 Skalering av datasett og resultater

Før vi går videre å se på et uavhengig testsett kan det være nyttig å ta en nærmere titt på resultatene valideringssettet oppnådde. Forecasting som tidligere beskrevet klassifiserer ikke data, men sikter heller på å predikere en tilnærmet numerisk verdi for den fremtidige verdien. På grunn av dette kan man ikke benytte seg av presisjon som man ellers ville gjort ved prediksjon via klassifikasjon. Vi belager oss heller på nøkkelindikatorer (KPI) som tidligere beskrevet i teorikapittelet av rapporten. Når man ser på et resultat som en RMSE verdi eller MAPE er det viktig å ha et godt grunnlag for å tolke verdiene.

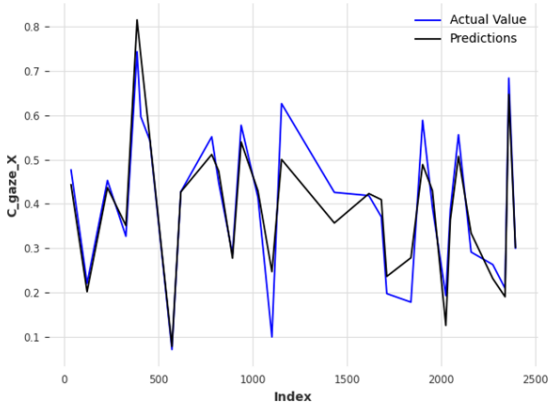
RMSE gir oss den gjennomsnittlige avstanden mellom de predikerte verdiene og de faktiske verdiene fra datasettet, så hva som kan tolkes som en god RMSE verdi kan variere. I et datasett hvor verdiene

varierer fra 0 til 1000 vil en RMSE på 0,8 være et svært godt resultat, men derimot hvis verdiene går fra 0 til 1 vil samme verdi ikke lenger være et godt resultat. Et RMSE resultat på 0.032 kan virke svært lavt, men da er det viktig å bemerke at alle verdiene i datasettene er skalert, og befinner seg nettopp mellom 0 og 1. Et resultat på 0.032 er fortsatt svært bra, men virker mindre absurd med dette i bakhodet. Ser man på Y-aksen til grafene ser man at verdien går fra 0 til 1, så med tanke på hvor tett opptil prediksjonen ligger den faktiske verdien i grafen vil dette resultatet være en tenkelig verdi.

For MAPE kan det være greit å holde et øye på prosenten ettersom man kan ende opp med en ufattelig høy verdi selv om prediksjonen tilsynelatende passer datasettet godt. Dette kommer av at MAPE deler den absolutte feilen på de faktiske dataene, så verdier nær 0 kan blåse opp MAPE resultatet betraktelig, noe man kan finne i et datasett med verdier mellom 0 og 1.

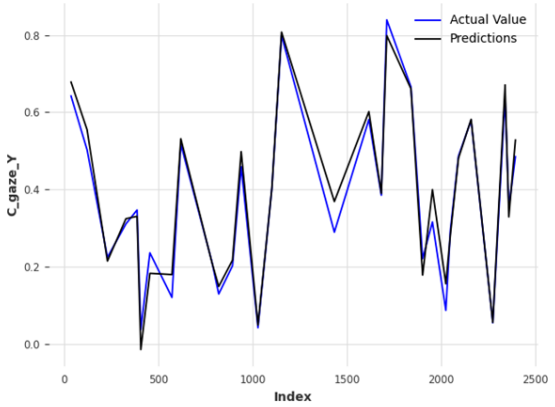
4.1.4.3 Forecasting av testsett

Figur 62: `C_Gace_X` vektor for uavhengig testsett



Forecasted- og den sanne verdi for blikkets x-vektor på uavhengig testsett

Figur 63: `C_Gace_Y` vektor



Forecasted- og faktisk verdi for blikkets y-vektor på uavhengig testsett

For videre testing av modellen ble modellen utprøvd på et uavhengig testsett. Scenen datasettet ble tatt fra var også en annen enn den modellen ble trent opp på for å se hvordan det ville påvirke prediksjonen. Datasettet bestod av 31 sakkader. Et langt lavere antall totale sakkader i valideringssettet, noe som kommer av at datasettet tatt i bruk for trening ble satt opp for å spesifikk fremprovosere mange sakkader, samt scenen ble kjørt langt lenger. Figur 62 og 63 viser tilsvarende grafer som beskrevet i avsnittet over. Modellen predikerte med en root mean squared error verdi på 0.044, samt en mean absolute percentage error på 0.15%. Dette er noe høyere verdier enn på valideringssettet, samt man ser enkelt større avvik i grafene mellom prediksjon og det faktiske datasettet, men det er å forvente. Resultatene viser fortsatt at modellen er i stand til å predikere sakkadiske bevegelser på et svært tilfredsstillende nivå.

Tabell 23 viser en oversikt over kjøringer gjort på alle datasettene som gruppen hadde samlet inn. Resultatene verdiene for RMSE og MAPE er en gjennomsnittsverdi på for alle datasettene innenfor en gitt scene. Modellen gir overordnet gode resultater for de fleste av scenegruppene. Dette viser til at modellen sin ytelse ikke er bundet til kjennskap til en scene. Det er dog enkelte scener som smoothmovement-still og mov med litt høyere KPI verdier en andre scener. Potensielle grunner til disse tilfellene vil bli diskutert i diskusjonskapittelet av rapporten. Men generelt predikerer modellen på samme nivå på de fleste scener som den gjorde på testsettet.

Tabell 23: Forecast prediksjon av alle scener

Scene	RMSE	MAPE
eyedata1	0.048	0.11%
eyedata2	0.068	0.12%
eyedata3	0.046	0.11%
lookandremember	0.044	0.17%
many-balls-disapear-1	0.042	0.013%
many-balls-disapear-2	0.045	0.11%
many-balls-disapear-3	0.044	0.17%
many-balls-disapear-4	0.042	0.10%
many-balls-still-1	0.044	0.14%
many-balls-still-2	0.040	0.11%
many-balls-still-3	0.041	0.11%
many-balls-still-4	0.040	0.15%
one-ball-move-1	0.048	0.16%
one-ball-move-2	0.034	0.05%
one-ball-move-3	0.059	0.30%
one-ball-move-4	0.050	0.13%
one-ball-still-0	0.032	0.07%
one-ball-still-1	0.057	0.14%
one-ball-still-2	0.045	0.11%
one-ball-still-3	0.045	0.31%
one-ball-still-4	0.046	0.12%
smoothmovement-mov	0.060	0.14%
smoothmovement-still	0.065	0.17%
smooth-pursuit	0.060	0.28%
taxitour-mov	0.047	0.07%
taxitour-still	0.050	0.13%

Gjennomsnitts RMSE og MAPE resultater for hvert datasett av en scene

4.2 Administrative resultater

Dette kapitelet vil gjøre rede for de administrative resultatene til prosjektarbeidet. Her inngår hvordan det er gjennomført, hvordan tidsforbruk har vært, og eventuelle utfordringer gruppen har erfart. Hensikten med dette kapitelet er å være til støtte for eventuelt videre arbeid av dette prosjektet, eller bidra til innsikt dersom en skal gjennomføre lignende prosjekter.

4.2.1 Fremdriftsplan

Prosessen ble i starten av prosjektet delt opp i fem ulike faser, hvor gruppen var innforstått med at disse kom til å endres på underveis i arbeidet. Å fastsette datoer i et slikt prosjekt er vanskelig, men det viktigste en slik plan har bidratt til er å definere forventete arbeidsoppgaver og problemstillinger tidlig i prosjektet. Likevel viste det seg at målsetningene og arbeidsoppgavene skulle være ganske nøyaktige for hva som ville dukke opp i løpet av prosjektet. Den største forskjellen mellom planlagt progresjon, og faktisk progresjon, kom i form av at man aldri var helt ferdig med en arbeidsoppgave. I fremdriftsplanen definerte vi 'Datainnsamling- og behandling' til å være fase nummer to, som skulle være ferdig 4. mars. Dette ble feil tilnærming, og var både en svakhet med gruppens tilnærming, og en begrensning med nytten til å sette slike delmål. Som presentert gjennom hele denne rapporten, har det naturligvis vært nødvendig å samle inn ny data etter behov. Dette kommer både som en konsekvens av manglende datagrunnlag for tilstrekkelig mengde treningsdata, men også som følge av at nye behov gjør at dataen som er samlet inn ikke er egnet for det spesifikke tilfellet. En måte å oppsummere dette på, er at selv om oppstartdatoen til

de forskjellige fasene har vært ganske nøyaktige, vil det være arbeidsoppgavene fra de forskjellige fasene helt frem til prosjektslutt.

4.2.2 Tidsforbruk

Målsetningen til gruppen var å holde arbeidsdager på 8 timer jevnt over hele prosjektarbeidet, men at både halve dager eller lengre dager kunne bli nødvendig ved behov. Utgangspunktet var at gruppemedlemmene ønsket å jobbe i felleskap så langt det lot seg gjøre, selv om det var åpent for å jobbe selvstendig dersom en hadde noen spesielle arbeidsoppgaver man ønsket å se på. Resultatet av dette er at gruppemedlemmene i all hovedsak har jobbet sammen, men at en fem prosent (om lag 20 til 25 timer) av tidsforbruket til gruppemedlemmene har godt til selvstudium.

Gruppemedlemmene har hatt prosjektarbeid og eksamen i samme fellesfag, samt at de har sine respektive deltidsjobber og andre forpliktelser. I den grad det har latt seg gjøre har hverdagen blitt lagt opp slik at gruppemedlemmene har hatt tilgjengelig tid på samme tidspunkt av uken. Resultatet ble at gruppen ønsket å benytte to og en halv dag i uken på starten av prosjektet, tre til fire dager i uken etter eksamen, og hele arbeidsuker etter påske. Etter behov var det aktuelt å legge inn helgearbeid, eller arbeid i ferier. I timelistene er disse periodene med ekstra arbeid reflektert. Resultatet er en arbeidskurve som er jevnt stigende gjennom hele prosjektarbeidet [69]. Totalt sett har gruppens medlemmer brukt 450 arbeidstimer hver.

Å beskrive hvor mye tid som går til forskjellig typer arbeid i et slikt prosjekt er noe vanskelig. Gruppen har overordnet registrert arbeidsoppgaver sammen med antall timer for dagene som har blitt jobbet. En utfordring med dette er at forskning og utvikling i et slik prosjekt ikke alltid er enkelt å skille. Man gjør valg i utvikling av statistiske metoder på bakgrunn av forskningen man gjør underveis. På samme måte så kan utvikling og testing av nye eksperimenter sies å være forskning. Gruppen har likevel prøvd å være strenge med seg selv slik at det kan presenteres et grovt bilde på hva en kan forvente å disponere tiden sin på i et slikt prosjektarbeid [68]. Gruppen har brukt om lag 30 prosent av tiden sin på dokumentasjon, mens om lag 30 prosent av tiden har gått til implementasjon av nye scener og maskinlæringsalgoritmer. De resterende 40 prosentene har gått til analyse, litteraturstudium og datainnhenting. Vi ønsker likevel å presisere at dette er svært individuelt og tilfellebasert, og vil variere ut ifra hvordan respektive grupper definerer de forskjellige kategoriene selv.

5 Diskusjon

5.1 Evaluering av øyedata

Etter første datainnsamling ble det åpenbart at det var hensiktsmessig å samle inn mer data for å ta en grundigere gjennomgang av de forskjellige scenene og algoritmene. De tidligere gjennomførte analysene av øyedataen var hensiktsmessig, spesielt med tanke på den begrensede tilgangen på tilstrekkelig øyedata til Moan, Nygaard og Ramberg. Det ga en god indikasjon på likhetene mellom de forskjellige algoritmene til manuelt klassifisert data. Når algoritmene nå skulle benyttes for å avgjøre hva som potensielt skaper usikkerhet og støy i dataen, viste det seg likevel nødvendig med et større datagrunnlag.

Fra analysen ble det klart et par ting som måtte tas i betraktning ved videre utvikling av nye scener. Først og fremst var det helt nødvendig å begrense støy, slik at forventede like resultater på tvers av scener ble forutsigbare. At smooth-movement still og move fikk så ulike resultater, til tross for at fikseringspunktet var det samme i begge scenene, tydet på at det var behov for å generalisere disse testene. Ved enkel visualisering av forskjellene mellom eksperimenter gjennomført med hodet i ro, sammenlignet med hodet i bevegelse gjorde det åpenbart for gruppen at etterprøvbare eksperimenter nødvendigvis måtte kjøres med minst mulig bevegelse fra testpersonene. Alternativet her var å benytte oss av støyreduserende algoritmer. Som beskrevet, var resultatene fra utprøvning av slike metoder ikke tilfredsstillende for bruk i dette prosjektet. 90hz er rett og slett ikke høy nok oppløsning for å fjerne støy , og samtidig beholde brukbare, nøyaktige målinger av øyedata. Å be testpersoner holde hodet i ro var noe gruppen i utgangspunktet ønsket å unngå, da en av de store fordelene med virtuelle omgivelser er hvor mobil brukeren er i miljøet. Ved å be brukeren holde seg i ro, blir dette aspektet ikke lenger en faktor. Tilsynelatende kan det se ut til at dette eliminerer fordelene med å benytte seg VR-hodesett til slike eksperimenter, sammenlignet med andre tradisjonelle innsamlingsmetoder. Gruppen ønsker derimot å argumentere mot denne tilnærmingen, på bakgrunn av spesielt to aspekter. Den første umiddelbare fordelen med innsamling i VR, er hvor stor kontroll utviklere har over designet av scenene. Som brukeren i VR er hele synsfeltet inne i det kontrollerte brukermiljøet som er designet. Dette betyr at utviklere har full kontroll over synsfeltet til brukeren under eksperimentet, og kan ta hensyn til både hodets og blikkets posisjon i sanntid. Den andre er at selv om dagens utstyr ikke nødvendigvis klarer å tilstrekkelig begrense støy, går utviklingen på området svært raskt. Det å lage skalerbare løsninger, og utforske bruken tidlig i utviklingen vil derfor fortsatt være svært verdifullt. Prisendringer for kraftig VR-utstyr tyder på at dette er teknologi som over tid kan bli tilgjengelig for flere og flere. Konsekvensen av dette blir drøftet ytterligere senere i denne delen.

Samtidig som det var store forskjeller i scenene, var det også merkbare forskjeller på tvers av algoritmene, selv for de samme scenene. Sammenligning på tvers av scener innebærer risiko da det kan være forskjeller knyttet opp mot den spesifikke datainnsamlingen for et eksperiment. Sammenligning på tvers av algoritmer byr på mindre usikkerhet, da alle algoritmene klassifiserer nøyaktig samme data. Dersom det var en fast fasit man kunne kommet frem til med øyedataen fra hodesettet, vil en kun anta at det var stor grad av likhet på tvers av algoritmene. Fra før av visste vi fra tidligere rapportering at I-VT mest sannsynlig kom til å klassifisere langt færre sakkader, avhengig av grensefarten som defineres. Det var derimot større forskjeller på algoritmene generelt enn først antatt. Dette er naturligvis delvis en konsekvens av forskjellige implementasjonsdetaljer. Som presentert har algoritmene forskjellige brukerdefinerte parametere, som kan tilpasses avhengig av hvordan de skal benyttes. Det som en kan ta ut fra dette er at det ikke er en klar fasit på hvordan en skal klassifisere øyebevegelser, dette underbygges av litteraturen. Et annet poeng som er verdt å tenke over, dersom man skal benytte maskinlæring til å klassifisere øyebevegelser spesifikt, er nettopp det at mangelen på en klar fasit som kan definere fasitdata som en maksinlæringsmodell kan trene på. Paradoksalt hadde en slik klar "beste" algoritme for å klassifisere fasitdata delvis eliminert behovet for en maskinlæringsmodell til å begynne med. Bruksområdet ville i så fall vært til sanntidsklassifikasjon, dersom en slik modell klassifiserer raskere enn tilsvarende algoritmer.

For å få en oversikt over nettopp denne usikkerheten, ble dette visualisert både med tanke på vinkelfart, samt romlig spredning og blikkposisjon generelt. Det var stor enighet om sakkader rundt

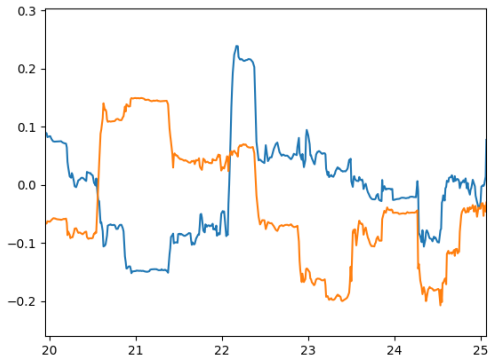
punkter hvor blikket åpenbart hadde beveget seg og vinkelhastigheten hadde sin amplitude. På samme måte var fikseringer enkle å definere rundt punkter hvor blikket holdt seg i ro over lengre tid. De store uenighetene kom i områder hvor sakkader startet og sluttet, samt hvor vinkelhastigheten var i endring, men hadde relativt lav hastighet. På bakgrunn av dette var det nødvendig å legge inn en ekstra dimensjon i den kvalitative analysen, hvor den romlige spredningen ble tatt til betraktning. Dette ble kombinert med å konkret gå inn å se på sammenhengen mellom forskjellige features, ved hjelp av feature extraction. Det viste, som teorien foreslår, at den romlige spredningen er en sentral del av hva som skiller på de forskjellige øyebevegelsene. Siden den romlige spredningen baserer seg på endring i posisjon over tid, tydet dette også på at de konkrete koordinatene til blikkets posisjon, også kunne benyttes i slik analyse.

5.2 Scener

For gruppen var det klart at dersom det skulle være mulig å benytte seg av øyedataen i videre analyse, så var man avhengig av metoder for å automatisk klassifisere data. Dette er i motsetning til manuelt klassifiserte data, som ikke bare er svært tidkrevende, men også utsettes for risiko for feil klassifisering på bakgrunn av den gruppens manglende profesjonelle kompetanse på øyets fysiologi. Visualiseringen i tre dimensjoner gir et potensielt verktøy som kan brukes til å i større grad evaluere og klassifisere data. Likevel er dette fortsatt ikke en oppnåelig oppgave for gruppen på dette tidspunktet. Dalveren og Cagiltay argumenter for at maskinklassifisering av øyedata er den eneste hensiktsmessige måten å evaluere øyebevegelser[35]. Som vist og diskutert, er det fortsatt svært store ulikheter knyttet til forskjellige klassifiseringsalgoritmer, og dermed vanskelig å si med sikkerhet at klassifiseringene er riktige. Avhengig av hvordan øyedataen skal benyttes, er ikke dette nødvendigvis et stort problem. Dersom klassifisert data skal benyttes i trening av en maskinlæring, er man avhengig av at denne dataen representerer en fasit. Dersom algoritmene allerede representerer en fasit, vil behovet for en maskinlæringsmodell være betraktelig mindre, da det allerede vil eksistere en sikker måte å vurdere øyedataen.

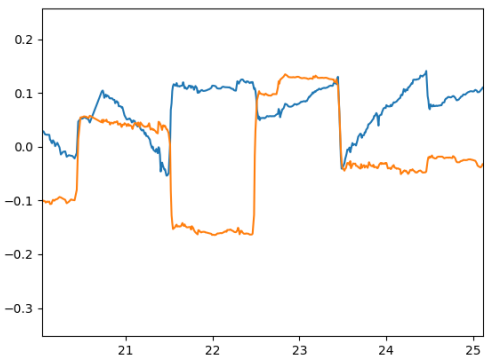
Øyebevegelser, og da spesielt ballistiske sakkader kommer som en konsekvens av det visuelle inputet personen skal prosessere. Det ble derfor laget scener som spesifikt prøver å fremprovosere forskjellige typer bevegelsesmønster og øyebevegelser. Disse kan benyttes spesielt på to områder. Analyse av data hvor man vet med stor grad av sikkerhet hva testpersonen kommer til å fiksere på, og som konsekvens klare forventninger til hvordan øyet kommer til å reagere. Den andre er at dersom man aksepterer at forskjellige typer visuelle stimuli sier noe om en person sin oppmerksomhet, vil man kunne benytte seg av ferdigklassifisert data i et ikke-trivielt klassifiseringsproblem. Dette vil da gå ut på om en kan benytte seg av maskinlæring til å oppdage forskjeller i fokus til personene som får sine øyebevegelser registrert i VR.

Figur 64: Many-Ball-Disappear x og y



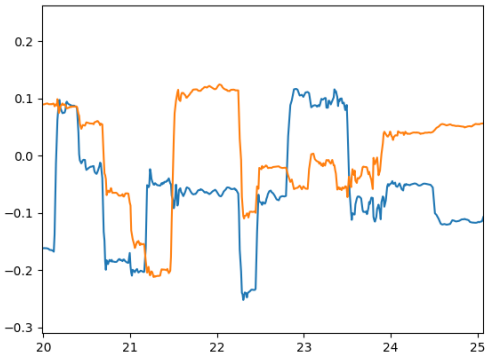
Blå: blikkets x-posisjon, Oransje: blikkets y-posisjon

Figur 65: One-Ball-Move x og y



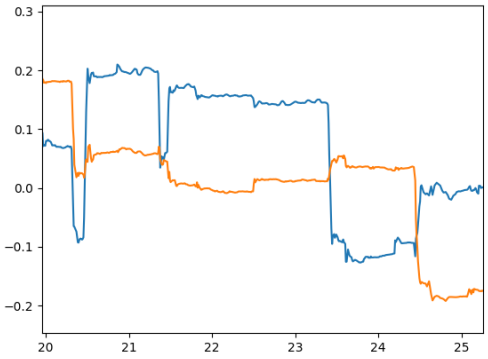
Blå: blikkets x-posisjon, Oransje: blikkets y-posisjon

Figur 66: Many-Ball-Still x og y



Blå: blikkets x-posisjon, Oransje: blikkets y-posisjon

Figur 67: One-Ball-Still x og y



Blå: blikkets x-posisjon, Oransje: blikkets y-posisjon

I hvor stor grad scenene sier noe verdifullt om en person sin oppmerksomhet, er vanskelig å konkludere med, og for øvrig også utenfor omfanget til oppgaven. Det som likevel er mulig å si kun ved å se på bevegelsen til øyet i x- og y-posisjon over tid, er at scenene lykkes med å fremprovosere forskjellige typer øyebevegelser. Den mest tydelige forskjellen er de store ulikhetene mellom scenene som har et enkelt definert fikseringspunkt, og scenene med flere potensielle objekter å feste blikket på. Det at disse ulikhetene i så stor grad manifesterer seg i øyedataen er en indikasjon på at deler av fremgangsmåten for å bygge scenene er riktig. Gitt at man har forkunnskaper om scenene, og hensikten bak disse er det også mulig å dedusere seg frem til hvilke scener som øyedataen representerer, selv der det er flere likheter. Mellom One-ball-still, og one-ball-move, er det langs x-aksen man kan se øyet beveger seg i en jevn bevegelse. Dette ansees som bevegelsen jevn forflytning. Mellom many-ball-still har punkter hvor en kan se blikket er festet på et punkt over en lengre periode, dette kan man ikke se i many-ball-disapear, da slike punkter blir borte i det blikket treffer objektet. Denne forskjellen kan argumenteres for er vanskeligere å skille enn mellom de to foregående scenen. Det kan likvel tyde på at vi derfor lykkes med å tvinge frem ballistiske sakkader, som testpersonen ikke har kontroll på selv. Many-ball-disapear sammen med many-ball still, er scene hvor man i minst mulig grad kan predikere hvor blikket kommer til å forflytte seg. Derimot har many-ball-disappear kanskje det største potensiale for å finne forskjeller i personers blikk som følge av de ballistiske bevegelsene den fremprovoserer. Sådan vil en scene med et fikseringspunkt som umiddelbart endrer posisjon til et bestemt punkt i rommet, være en potensiell scene hvor slike bevegelser kan oppstå, samtidig som de er mer forutsigbare.

Scenene er laget slik at testpersonellet skal ha så stor kontroll over testscenarioene som mulig. Vi anså dette som helt nødvendig, da vi av erfaring vet at man aldri helt sikkert kan si at datagrunnlaget egner seg til en gitt oppgave før man har begynt å jobbe på den. Et eksempel på dette er hvordan det å ha et tilfeldig tidsrom mellom hver oppdatering av fikseringspunktene sin posisjon i scenen virker som det mest gunstige for alle tilfeller. Fra maskinlæringsteorien, og egne erfaringer, er det som regel lurt å ha et datasett som har mye variasjon og lite likheter seg mellom når disse skal benyttes i trening. Dette gir mening for å blant annet unngå overfitting, hvor modellen trener på mønstre som ikke generaliserer de ønskede attributtene vi ønsker. Et slikt tilfelle vil være et typisk område for bekymring når man trener opp en modell på temporal data, da du ikke ønsker at klassifikasjonene skal skje på bakgrunn av et mønster du har programmert inn i scenen. Et aspekt som vi derimot ikke tok til betraktning i den originale analysen, var hvordan faste tidsintervaller for oppdatering kunne benyttes til å enkelt finne frem til tidsintervall hvor vi med sikkerhet visste at en sakkadisk bevegelse ville finne sted. Dette er naturligvis et nyttig verktøy dersom man skulle ønske å automatisk klassifisere øyebevegelser. Dersom du vet at ballen testpersonen ser på vil bevege seg nøyaktig hvert sekund, vil du enkelt kunne filtrere ut disse intervallene til bruk under trening. Samtidig er dette et nyttig verktøy til prediksjon ved hjelp av forcasting. Her er en helt avhengig av noen fastsatte parametere for å kunne predikere øyets bevegelse over tid. Det vil være umulig å predikere hvor øyet kommer til å bevege seg i løpet av de neste to sekundene, dersom det er mulighet for at fikseringspunktet til personen endrer seg på et vilkårlig tidspunkt i løpet av

dette intervallet. Denne forskjellen kom tydelig frem da arbeidet med forcasting startet.

Som nevnt i rapporten er det lagt til rette for at eksperimentene skal kjøre et deterministisk seed, som er etterprøvbart og kan kjøres på nytt. Hensikten med dette er at en skal kunne kjøre et eksperiment hvor oppdateringsfrekvens samt eventuell hastighet på objektene i scenene er tilfeldig, samtidig som at testpersonell skal kunne kjøre scenen med de samme parameterne på nytt. Vi anså det som nyttig å ha parametere som tilfeldig oppdaterte seg i løpet av et eksperiment, som vi også kunne kjøre på nytt med nye testpersoner for å sammenligne, eller med samme testperson dersom det var behov for dette. Dette er en oppfatning gruppen fortsatt sitter med, men vi har erfart at det er et par potensielle feilkilder som dukker opp som følge av denne tilnærming til eksperimentdesign. For det første så er det svært nyttig å kunne samle inn data fra samme randomiserte scener fra forskjellige testpersoner. Samme scene ble kjørt på fem minutter, med samme random seed for alle tre deltagere. Dersom målsetningen er å skille mellom testpersoner, er dette et godt utgangspunkt. På denne måten eliminerer man potensiell påvirkning av ulikheter i scener, og ser kun på hvordan øyet reagerer på samme visuell input. Dersom målsetningen er å skille på visuell input, skal man være varsom for at modellen trenes på intervaller som eksisterer en person har sett på, og valideres på det samme intervallet som eksisterer i et annet datasett. Her er det nødvendig å filtrere ut dataen slik at et tidsintervall fra et seed som benyttes i treningen ikke benyttes i validering eller testing. På denne måten kan man forsikre seg om at valideringspresisjon ikke blir kunstig høy, og representerer modellens presisjon på helt usett datasett. En måte å oppnå dette på er å skille ut valideringssettet før dataen deles inn i sekvenser og stokkes om. Slik unngår man at samme tidsintervall på samme seed i ulike datasett, benyttes i både trening og validering. Samtidig kan en bekrefte eller avkrefte eventuell påvirkning av eventuell feilaktig trening, ved å teste på et helt uavhengig datasett, med et seed som aldri benyttes i verken trening eller validering. Dette ble gjort for samtlige forsøk i dette prosjektet. I tillegg bidrar en teknikk som KFolding til å underbygge troverdigheten til disse resultatene. Gjennomsnittlig resultater opp mot trening- og valideringspresisjon, samt relativt lave standardavvik bidrar videre til at testresultatene vurderes som gode og valide.

På generell basis så er det mange faktorer som skal hensyntas når en både lager og benytter seg av scener til øyedata. Både for manuell evaluering og benyttelse av maskinlæring. Dette er en øvelse som både er vanskelig og tidkrevende, men med rett fremgangsmåte er scenene svært brukbare og skalerbare. Både med tanke på hvordan scenene kan videreutvikles, og hva en kan anvende den innhentede dataen til.

5.3 Konvolusjonelle nevrale nettverk vs Recurrent Nevrale Nettverk

En utfordring når man jobber med sekvensiell data i konvolusjonelle nevrale nettverk, er å representere hvordan endringer går over tid. Spesielt i dette tilfellet hvor blant annet antall klassifiseringer av sakkader og fikseringer viser seg å være en egenskap man kan bruke til å klassifisere scener. Det er også gunstig å se hvor blikket "hviler" for å skille mellom de forskjellige scenene, da enkelte scener lar brukeren fiksere på et punkt over tid, mens andre ikke tillater dette. Den romlige spredningen over tid representeres av x og y posisjonen i rommet. Selv om det allerede er vist mulig å se forskjeller mellom scenene fra bildene som genereres, betyr ikke dette nødvendigvis at de er godt egnet for å trenes i en modell. Den kanskje største svakheten i måtene dataen preprosesseres nå, er hvor stor del av bildet som ikke har noen nyttig informasjon for treningen. Dersom man ønsker å representere den romlige spredningen mer nøyaktig, med absolutte x- og y-verdier på plottene, vil det for mye av dataen bli enda større tomrom uten relevant data. Selv med en relativt god presisjon på selv helt uavhengige testsett som er betraktelig bedre enn vilkårlig gjetting, er det fortsatt ikke høyt nok til å kunne klassifisere med stor grad av sikkerhet.

Det er tilnærmet uendelig mange måter å representere øyedataen billedlig for å bruk til trening og validering. Samtidig er feature engineering en vanskelig og tidkrevende prosess. Det er lagt til rette for å kunne sammenligne relevante attributter for å skille på de forskjellige scenene med feature extraction, men selv med gode indikasjoner på hvilke features som egner seg til trening, er det å finne egnede metoder for å representere dette utfordrende. Dette oppleves som en av de største ulempene med å jobb med øyedata i CNN. Fra våre resultater er det øyedata på tre sekunders intervaller som gir de beste og mest troverdige resultatene. I seg selv er det å få høy presisjon for klassifikasjon etter kun tre sekunder med øyedata et hensiktsmessig resultat. Den begrensende

faktoren vil i så fall være innsamling av data. Vår modell ble trent opp på et datagrunnlag med over en time med øyedata. Med tre sekunders lange intervaller fikk vi 1605 unike bilder, hvor ingen av datapunktene ble gjenbrukt. Deler man opp dette i et forhold på 60, 20, 20 får man i underkant av 1000 bilder å trene på. For å få tilstrekkelig høy presisjon er det vanligvis nødvendig med betraktelig mer data enn dette. Bildeaugmentering er en vanlig metode å oppnå dette. Hvor mye tid som er hensiktsmessig å benytte på dette er usikkert, da resultatene til RNN modellen virket å gi betraktelig bedre resultater. Om man likevel bestemmer seg for å øke dette til eksempelvis 400, eller 500 datapunkter per klassifikasjon, kan man oppnå enda litt høyere presisjon.

En av fordelene med å benytte RNN for klassifikasjon av øyedata, er at en ikke trenger å gjøre nevneverdig preprosessering av dataen før den kan benyttes i treningen. På samme måte som med CNN er det nødvendig å gjøre tilstrekkelig feature selection for å kunne ta gode valg av hvilke data man ønsker å benytte i trening og validering. En annen stor fordel med RNN er at man trenger betraktelig mindre øyedata for å klassifisere med akseptabel presisjon, selv på uavhengig testsett. Som med språklig data, er det mulig å sette et intervall mellom eksempelvis 50-200 datapunkter, som omtrent tilsvarer mellom et halvt og to sekunder. Dette gir betraktelig bedre presisjon enn CNN modellen, som i tillegg var avhengig av mer data per bildet.

Ved å trene på variabel inputlengde, vil også modellen bli mer anvendbar i praktiske eksempler. Avhengig av de ønskede implementasjonsdetaljene til utvikleren, kan eksperimentene tilpasses etter behov. Dersom det viser seg at et eksperiment er avhengig av å ta inn flere enn datapunkter som følge av designet på nevnte eksperiment, vil modellen i mye større grad være fleksibel med inputlengde. Siden modellen i all hovedsak trener på rådata er den svært anvendbar for klassifiseringer i sanntid. Avhengig av implementasjonen trenger en kun enkle beregninger som øyets vinkelhastighet for å klassifisere data. Dette lar en både teste og benytte hvordan modellen presterer i praksis, og gjør at modellen også får flere potensielle bruksområder.

5.4 Resultater og potensielle feilkilder i RNN

Gruppens resultater for både klassifikasjon av scener og testpersoner var svært tilfredsstillende. Det er i løpet av denne rapporten blitt presentert tidligere arbeid med maskinlæring på øyedata, disse deler en del likhetstrekk med vårt arbeid. Erfaringer delt i disse arbeidene har vært utslagsgivende for resultatene som er oppnådd i dette prosjektet. Det var åpenbart for gruppen at for å kunne benytte seg av maskinlæring i dette prosjektet så var man avhengig av en klar fasit for å vurdere resultatene. Å klassifisere forskjellig type stimuli ved bruk av øyedata er en av problemstillingene som ikke er ukjent fra litteraturen, og er motivasjonen bak utviklingen av de nye scenene. Et aspekt som skiller vårt arbeid fra annet tidligere arbeid, er hvordan disse scenene er laget for å nettopp reflektere forskjellige typer øyebevegelser. Hvor vinklingen på annen litteratur overordnet kan beskrives som studier om hvordan øyet oppfører seg annerledes gitt ulike typer stimuli, er gruppens forskning gjort med antagelsen om at vi vet hvordan med rimelig sikkerhet hvordan øyet kommer til å reagere. Dette er gjort mulig basert på grunnlaget lagt i Moan, Nygaard og Ramberg sitt arbeid. Som en konsekvens av dette var gikk derfor initielt arbeid med maskinlæring ut på å se om en modell kunne se oppdage og lære av disse forskjellen gitt den tilgjengelige øyedataen. Resultatene gruppen oppnådde indikerte et stort potensiale på dette området.

Fra resultatene ser en tydelig at det er mulig å skille mellom forskjellige typer scener med høy grad av presisjon. Derimot er det noen spesifikke hensyn som må bli tatt til betraktning. Many-ball-disappear og many-ball-still var tilsynelatende for like for at modellen kunne skille disse med tilstrekkelig presisjon. Dette er ikke helt unaturlig. Øyets posisjon over tid, samt vinkelhastigheten var verdier som modellen benyttet under treningen. Mens one-ball-still og one-ball-move hadde veldig klare og definerte punkter i scenen som testpersonene skulle fiksere på, var scenene med flere baller i stor grad avhengig av testpersonens tilnærming til testen. Det er mulig å argumentere for at many-ball-disappear gjør en bedre jobb med å tvinge brukeren til å gjøre en spesifikk type øyebevegelse, selv om blikkets videre bevegelser etter ballene forsvinner ikke er bestemt av hvordan scenen implementeres av utvikleren. Det en kan se fra øyedataen er at denne scenen lykkes i å tvinge frem endring i blikket hos brukeren. På den andre siden er many-ball-still svært uforutsigbar. Denne vil kun være avhengig av hvordan brukeren velger å flytte blikket sitt innad i scenen. For eksempel kan testpersonen velge å ha fokus på en og en ball, slik at blikket vil ha tilsvarende bevegelser

som i one-ball-still. Dersom testeren velger å se på så mange baller som mulig, kan det ligne mer på en one-ball-move, eller many-ball-disappear. En konklusjon som er mulig å dra fra dette er at modellen ikke nødvendigvis klarer å skille på forskjellig type stimuli, da many-ball-still ikke med samme nøyaktighet kunne klassifiseres som de tre andre scenene. Mange baller på skjermen vil ikke nødvendigvis fremprovosere en spesiell type adferd, og er også vanskeligere å kontrollere ved utførelse av eksperimenter.

Et svært viktig poeng som ble diskutert av gruppen underveis i prosessen, var hvorvidt det var forsvarlig å fjerne many-ball-still på bakgrunn av usikkerheten knyttet til klassifisering av scenen. Det er naturligvis ikke en hensiktsmessig praksis å vende et blindt øye til problemer som ikke gir ønskede resultater. Det er derfor viktig å presisere at denne avgjørelsen ble tatt for å utforske modellen sin adferd når en scene som potensielt kunne virke mot sin hensikt ble fjernet. Hele forskningsprosessen har blitt designet av gruppen med bakgrunn i erfaringer gjort av Moan, Nygaard. Dette inkludering design, implementasjon og datainnhenting ved hjelp av scenene. I løpet av et prosjektarbeid som dette er det naturlig å finne mangler og forbedringspotensialet i eget arbeid. Å være skeptisk til eget arbeid er essensielt, da resultatet av arbeidet ikke nødvendigvis blir det en selv forventet. Motivasjonen bak å utforske resultater uten many-ball-still, kom fra nettopp dette ønsket om videre innsikt i kapabiliteten til maskinlæring sammen med øyedata fra HMD-et. Konsekvensen av dette ble en mye høyere presisjon, og mye lavere usikkerhet selv med spesifikke scener satt opp mot hverandre. Like viktig var at den markante forskjellen ledet gruppen til tanken om at en var helt avhengig av å designe scener gir klare forskjeller i øyets adferd for å oppnå ønsket presisjon. Samtidig som at behovet for å begrense tilfeldigheter knyttet opp mot brukerens tilnærming til eksperimentet var svært viktig.

Disse betraktningene var et viktig utgangspunkt i vurdering av hvordan en kunne benytte maskinlæring for å skille på forskjeller gitt den samme visuelle inputene. Ved å benytte seg av de samme scenene, kan en utforske hvorvidt modellen kan klare å klassifisere hvem som har på seg hodesettet basert på brukerens øyebevegelser. I motsetning til klassifisering av scener, er det her absolutt ønskelig at personer skal benytte de samme seedene gjennom trening og validering. På denne måten kan man isolere ulikhetene i brukerens blikk, i stedet for at den skal trene på hvilket spesifikke seed som brukeren har sett på under datainnsamlingen. Presisjonen og loss fra resultatene her var også positive. Selv med trening, validering og testing på tvers av alle scenene ga den en presisjon på opp mot 70 prosent på helt uavhengig data. Selv om dette er langt under en presisjon man kan belage seg på for konsekvent riktige klassifikasjoner, er det likevel et resultat som tyder på distinkte forskjeller i personer sin øyebevegelse på tvers av alle scenen.

Presisjonen steg betraktelig når man begrenset trening og testingen til kun å se på en og en scene. Many-ball-disappear lot oss klassifisere riktig person i godt over 90 prosent av tilfellene. Spesielt interessant er det at nettopp denne scenen ga så gode resultater. For scener som one-ball-still eller one-ball-move, vil forskjeller til dels implisere at det er mulig å se forskjeller på personers fikseringer og jevne forflytninger. Det at presisjonen ble så høy for many-ball-disappear, gir utrykk for to mulige tilfeller som lar oss skille på spesifikke brukere sine blikk. Den første er at brukere har forskjellig adferd på øyebevegelser gitt raske oppdateringer som gjør at en ikke kan fiksere på et punkt. Dette kan for eksempel være hvilken retning, og hvor langt brukere velger å endre blikket innad i scenen for hver gang scenen oppdaterer seg. Den andre muligheten er om det er forskjeller på hvordan personer sine ballistiske sakkader registreres av eyetrackingen, gitt den samme visuelle inputen og den samme adferden. Akkurat denne muligheten kan være svært interessant å utforske videre på.

5.5 Klassifisering i sanntid

Klassifisering i sanntid bærer preg av fall i ytelse som kommer når modellen skal klassifisere øyedata. Som nevnt kommer dette av mangelen på asynkron klassifisering, slik at oppdateringen til selve scenen må vente på at klassifiseringen er ferdig. Dette gjør at eksperimentene over tid blir trege, og har uønsket adferd. Konsekvensen er at scenene ikke oppdateres og viser det som de er designet for, som også gjør at blikket til brukeren ikke ser på scener tilsvarende det modellen er trent på. Selv med dette problemet fungerer den nåværende implementasjonen godt nok til å vise at modellen også fungerer til å klassifisere på usett data, og bygger opp under konklusjonen om at testresultatene

er reelle. Selve sanntidsklassifikasjonen var litt utenfor oppgavens omfang, men uansett en naturlig videreutvikling av forskningen som er representert av gruppen. Samtidig var det faktum at det bidrar til å bekrefte resultatene fra trening og testing en god grunn for å få det på plass. Gruppen brukte ikke mer tid på å forbedre ytelsen i Unity, men dette er absolutt et område som ansees som hensiktsmessig å utforske ved videre arbeid.

5.6 Time Series Forecast

For å kunne trene opp og senere predikere sakkader er forecasting modellen avhengig av et klassifisert datasett. Fra den tidligere rapporten av Moan, Nygaard og Ramberg, samt gruppens egen evaluering av det tidligere arbeidet gjort ble det tydet at HMM er en av de bedre skikket algoritmene tilgjengelig til å klassifisere et datasett. Fra Tabell 4 i punkt 4.1.1. Datasett kan vi se at HMM er algoritmen som oftest klassifiserer en sakkade. Forecasting modellen er avhengig av store og mange nok datasett for å prestere optimalt. Som påpekt i punkt 5.1 var det uenighet i klassifisering mellom algoritmene i områdene rundt starten og slutte av en sakkade, samt i områder hvor vinkelhastigheten var i endring, men hastigheten var relativt lav. Noe som potensielt kan utspille seg ettersom HMM er ivrig til å klassifisere en sakkadde er at noen øyebevegelser eller datapunkter kan bli klassifisert som en sakkade uten at de nødvendigvis er det. Disse datasettene kan da ha en negativ påvirkning på treningen av modellen. Ellers hvis et slikt datasett blir tatt i bruk under testing kan prediksjonen bli dårligere enn sammenlignet med andre datasett, ettersom datapunktene i datasettet ikke nødvendigvis oppfører seg som en normal sakkade. Fra oversikten over gjennomsnittsverdier for prediksjonen over alle scener i tabell 23 i punkt 4.1.4.3 Forecasting av testsett er det enkelte scener som skiller seg ut resultatmessig. Eksempelvis smooth pursuit samt smoothmovment-mov og –still har prediksjoner med relativt større forskjeller enn andre scener. Med tanke på øyebevegelsene fremprovosert i disse bestemte scene er det en mulighet for at HMM potensielt har tolket en rask jevn forfølgelse bevegelse som en sakkade, og det er med på å forklare resultatene man ser fra kjøringen. Det samme kan argumenteres for taxitour datasettene da scenen ikke har et bestemt formål og ønsket øyebevegelse som de nye utviklede scenene. Testpersonen har mye frihet i scenen, og scenen sin natur kan skape mye tvetydighet i datapunktene. Det skal like vel bemerkes at det ikke nødvendigvis tar noe vekk fra resultatene oppnådd i denne sammenheng, ettersom forecasting ikke handlet om klassifisering av datasettene, men heller prediksjon for fremtidig bevegelse av gitt data. Hvis ved en senere anledning en bedre egnet klassifiseringsmetode skulle presentere seg selv skal man kunne trene opp forecasting modellen på det nye klassifiserte datasettet, og predikere sakkadiske bevegelser.

Når RMSE resultatene som ble presentert i resultatkapitelet ble utregnet var det de predikerte verdiene til de tre blikkdimensjonene X, Y og Z som ble kalkulert. Men det skal nevnes at N-BEATS modellen vil forcaste både vinkelhastighet og vinkelakselerasjon i tillegg til blikkvektorene i hvert punkt for datasettet. Ettersom det var plasseringen til det fremtidige blikket som var av interesse var det derfor kun resultatene for selve blikkvektorene, altså blikkets plassering, som ble utregnet og presentert i resultatkapittelet av rapporten. Det er likevel mulig å utregne nøkkelindikatorene for alle features predikert av modellen, enten enkeltvis eller sammen som en gruppe av verdier. Nøkkelindikatorverdiene blir da noe høyere. Eksempelvis på det uavhengige testsett ga modellen en prediksjon på vinkelhastighet, akselerasjon samt X, Y, Z vektorer med gjennomsnittlig RMSE verdi på 0.052, mot en RMSE verdi på 0.044 for bare X, Y og Z vektor for det samme datasettet. Dette er fortsatt gode resultater, men en markant prosentvis økning i usikkerhet tilknyttet prediksjonen fra å bare se på blikkvektorene.

Økning kan forklares ved den potensielle usikkerheten man finner i øyets vinkelakselerasjon og vinkelhastighet. Mens det er det faste mønsteret som muliggjør forecasting av sakkader, vil fortsatt sakkadene ha noe variasjon mellom seg. Spesifikt lengden på sakkaden. Som beskrevet i punkt 2.4.2.1-Sakkader så vil normalt en sakkade vare fra 30 opp mot 120 millisekunder. Den varierende lengden kan skape en usikkerhet når sakkaden først er i ferd med å avsluttes, og vinkelakselerasjon og hastighet verdien først begynner å synke. Denne initiale forandringen av verdiene i begynnelsen av sakkadens avslutningsfase kan potensielt være vanskelig for modellen å predikere ettersom det ikke nødvendigvis er noe i dataen som presist tilsier i hvilket datapunkt sakkaden begynner på avslutningsfasen. Dette kombinert med de varierende sakkade lengdene som resulterer i at det

ikke er et fast overordnet tidspunkt i et datasett hvor forandringen vil skje gjør prediksjonen av disse verdiene noe mer usikker. Featurene blir beholdt ettersom de kan være av interesse ved videre utvikling av prosjektet. Ved tilfeller av ulike medisinske tilstander vil oppførselen til en sakkade forandre seg. Eksempelvis kan tilfelle av Huntington sykdom resultere i at sakkder holder en lavere hastighet. Sakkader med dempet fart er ikke uvanlig i de tidlige fasene av Huntington. Det åpner muligheten for å se om man kan benytte seg av vinkelhastighet featuren for å oppdage irregulariteter i en person sakkader[5].

5.7 Datasett og datainnsamling

En helt sentral del av prosjektarbeidet, har vært den kontinuerlige innsamlingen av ny øyedata. På slutten av dette prosjektet sitter gruppen igjen med godt over to timer øyedata fra de fire nye eksperimentene som har blitt brukt til trening av modellene. Innsamling samt behandling av så store mengder data, er både tidkrevende og utfordrende. I innsamlingsfasene med frivillige testpersoner, samlet vi inn data i 30 sekunders intervaller per scene. På denne måten kunne man belage seg på å bruke omkring 3-4 minutter per testperson på datainnsamlingen på de nye eksperimentene. Eksperimentene er designet på for innsamling av spesielle typer bevegelser, og kan være slitsomme for øynene over tid, spesielt many-ball-disappear. Scenene som var designet fra det tidligere prosjektet kan sies å være mer visuelt interessante, og sådan enklere å være i over lengre perioder. Utfordringen er at dersom en skal skille mellom eksempelvis forskjellige testpersoner, er ikke 30 sekunder per scene på langt nær nok. Gruppen samlet først inn fem minutter per medlem av hver scene, som viste seg å ikke være nok. Etter enda en innsamling med fem minutter per scene, begynte man å se de gode resultatene som er presentert. Dette er den største begrensende faktoren for arbeidet som er presentert. For gruppemedlemmene som er interessenter i prosjektarbeidet og deres resultater, er det enklere å motivere seg for å sitte 20 minutter i strekk i relativt monotone eksperimenter av den grunn at det er strengt nødvendig. For frivillige testpersoner er ikke dette en realistisk forventning. Samtidig er det ikke utenkelig at slike lange intervaller med innsamling av data, kan gjøre at dataen blir av dårligere kvalitet etter hvert som testpersoner blir slitne av å se på det samme over lengre tid. Dette er ikke en problemstilling som er unik for dette prosjektet, snarere tvert imot. Å samle inn data er et stort prosjekt i seg selv, og datagrunnlaget gruppen presenterer er både mangfoldig og gjenbrukbart. All dataen er navngitt med navnet på scenen, seedet som er benyttet, og et alias for testeren. Dataen vil derfor kunne funke som et godt grunnlag for videre utvikling og utforskning av øyebevegeler og maskinlæringen, selv uten tilgang på utstyr for å samle inn mer data. Dersom det er ønskelig å utforske områder som krever at man samler inn mere data, er det derimot viktig å være klar over mengden som kommer til å bli nødvendig i analysen.

5.8 Videre forskningsarbeid

3D-Motion-Technologies har gitt utrykk for at forskningsarbeidet som gruppen har foretatt seg i dette prosjektet, kan bli brukt videre til blant annet å oppdage abnormaliteter i øyets bevegelser hos personer. På denne måten kan prosjektet potensielt være starten på arbeid med å fange opp sykdommer i tidlig stadium ved bruk av øyedata i VR. Dette er lenge frem i tid, men arbeidet og resultatene som er presentert gir et optimistisk utgangspunkt for videre forskning på dette området. For at dette skal bli en realitet, er det spesifikke hensyn som må bli tatt til betraktning.

Som presentert i forrige delkapittel, er det å samle inn data svært tidkrevende og til tider slitsomt, avhengig av eksperimentvarigheten. Logistikken med å samle inn data for å skille på friske øyne, og øyne som har avvikende adferd, kan tenkes å være svært utfordrende. De spesifikke hensynene som i dette tilfellet må bli gjort er utenfor omfanget til denne oppgaven, men gruppen kan likevel presentere forslag til å effektivisere dette arbeidet gitt erfaringer fra arbeidet som er gjort.

For det første så kan det å gjenkjenne typer scener og hvem som benytter HMD-et virke som relativt trivielle problemstillinger. Likevel er det viktig å se på hva resultatene kan implisere, og å vurdere hvorvidt de er skalerbare til andre problemer. Å kunne skille mellom hvem som benytter HMD-et kan være overførbart til andre forskningsspørsmål innenfor klassifikasjonsproblemet. En av disse kan være å klassifisere sykdommer i tidlig stadiet, men det å samle inn sensitiv øyedata

fra et stort antall pasienter er som nevnt krevende og omstendelig prosess. Gruppen vurderer det strengt nødvendig å utforske brukbarhetene på generelt grunnlag, før en går videre på en slik prosess. Dette kan innebære å forske på andre måter å klassifisere øyedataen enn kun etter scener og testpersoner, som vil være mer overførbare til klassifisering av sykdommer. Samtidig vil det kunne være nødvendig å optimalisere modellene til å behandle flere klasser og større mengder data, med data man har lett tilgjengelig. På denne måten vil man som forsker få større innsikt i hvor mye data en vil trenge å samle, og hvilke behov som eventuelt kan dukke opp med design av nye scener og eksperimenter. Er det en ting gruppen har erfart fra prosjektarbeidet, så er det at forskning som dette er en iterativ prosess, hvor man kontinuerlig lærer av erfaringer. Gruppen argumenterer derfor at det er mest forsvarlige blir å lage en så skalerbar løsning som mulig før man begynner datainnsamlingen fra pasienter. Dette tror gruppen vil begrense behovet for mange runder med tidkrevende innsamling av sensitiv data, som totalt sett vil være gunstig for både forskere og pasienter som bidrar.

Et konkret tiltak gruppen forslår, er å gjøre en større datainnsamling med flere enn tre testpersoner fordelt de nye scenene som er laget i dette prosjektet. Det er vist at many-ball-disappear kunne skille mellom testpersoner med svært høy presisjon, men med et begrenset datagrunnlag lot ikke dette seg gjøre for scenene one-ball-still og one-ball-move. Many-ball-disappear har variasjon knyttet opp mot brukerens reaksjon på ballene som forsvinner. I scenene one-ball-still og one-ball-move er det derimot nesten eksklusivt brukernes stille fiksering og jevne forfølgninger som vil skille øyedataen. Om dette lar seg gjøre vil det være et stort steg for å avgjøre om nøyaktigheten i eyetrackingen er tilstrekkelig for å skille på små detaljer i brukerens blikk. Hvis teknologien er nøyaktig nok antas det å kreve enda mer data enn per person for å gjennomføre en slik analyse. Den nåværende RNN-modellen er trent på omlag 10 minutter for hver deltager per scene, gruppen ser for seg at en dobling av dette datagrunnlaget vil gi et godt utgangspunkt. Selv om resultatene fra denne videreutviklingen av eksperimentet er utilfredsstillende, vil fortsatt metodikken være nyttig. Den raske utviklingen i VR-teknologi gjør at forbedring i eyetracking kan innsamling av øyedata mer egnet for dette domenet i en ikke alt for fjern fremtid. Sånn sett er det mulig arbeide med forskningen parallelt med utviklingen av teknologien.

5.9 Etiske hensyn

Når man jobber med forskning på områder som er så nye som eyetracking i VR, er det helt essensielt å være varsom og oppmerksom på innflytelsen arbeidet kan ha. Siden arbeidet gruppen gjør vil ha direkte innflytelse på 3D Motion Technologies sin videre forskning, er det også nødvendig å presentere etiske problemstillinger og hensyn som må bli tatt til betraktning ved videre arbeid.

For det første så har gruppen vist at det er mulig for en maskinlæringsmodell å gjenkjenne brukeren av et HMD-ed, kun basert på øyets bevegelse inne i en scene. Ved datainnsamling er dette er viktig poeng, og gruppen informerte testdeltagere om at slike parametere kunne være unike for brukeren. De svært nøyaktige klassifikasjonene gjør derimot slike hensyn enda mer sentrale i problemstillingen. Det er ikke ønskelig å lage et fingeravtrykk for personer basert på hvordan øyet beveger seg som følge av ulike visuelle stimuli. Dette er informasjon som potensielt kan misbrukes, og må behandles med varsomhet. Datainnsamling av store mengder øyedata fra enkeltpersoner må derfor bli sett på som sensitivt. For gruppens vedkommende, og de aktuelle frivillige testpersonene som har deltatt, så er ikke dette en problemstilling som er relevant til øyedataen som er samlet inn i dette prosjektet. Testpersoner har sjeldent mer enn et minutt data per scene, som ikke er nok til å identifisere hvem som har benyttet HMD-et. Gruppens medlemmer har mye større mengder, men som vist synker presisjonen etter hvert som man inkluderer forskjellige typer visuelle stimuli i treningen. Det kan derfor anse at dataen samlet inn ikke er tilstrekkelig for å lage en form for fingeravtrykk for aktuelle testpersoner for øyeblikket. Likevel vil man ikke kunne utelukke at utvikling av teknologi sammen med nye runder med datainnsamling kan ha denne effekten. Dataen må derfor behandles varsomt, og forskningen må være etisk forsvarlig.

Et annet poeng knyttet opp mot bruken av metodene som gruppen har presentert, er at en må ta høyde for feil i klassifisering. Det er ingen garanti for at gjenkjenning av testpersoner vil kunne skalere tilstrekkelig til identifikasjon av sykdommer i tidlige stadier. Datagrunnlaget må være stort, men selv med veldig høy presisjon skal man være veldig forsiktig med å benytte klassifiseringer gjort

av maskinlæringsmodeller som dette når en stiller en diagnose. Fokuset burde være på å kunne identifisere risiko for sykdommer hos pasienter. Samtidig må det spesifiserer at en slik modell både kan klassifisere sykdom som ikke er til stedet, og kunne feile i å identifisere sykdommer som pasienten innehar. Nøyaktigheten til slike modeller på dette domenet er umulig å forutsi, men prinsippene om varsomhet og kildekritikk vil uansett være de samme.

5.10 Prosjektarbeid og teamsamarbeid

Arbeidet med prosjektet har tidvis hvert utfordrende, men til enhver tid vært givende. Det har gått ned svært mange arbeidstimer til prosjektet, samtidig som alle gruppemedlemmene har hatt forskjellige forpliktelser som også er tidkrevende. Planlegging og tilrettelegging for en jevn progresjon har derfor vært viktig, og i så stor grad som mulig holder seg til 8 timers arbeidsøkter i ukedager. Gruppen har til dels lykkes med dette, men mot slutten ble det klart at det var nødvendig å skru opp arbeidsmengden for å få alt på plass avslutningsvis. Dette er delvis en konsekvens av at timeplanen var lagt opp til at undervisning og eksamener skulle gjennomføres tidlig i semesteret, slik at siste del av semesteret eksklusivt skulle brukes til bachelorarbeid. I sum har gruppen endt opp med et timeantall og en arbeidsmengde som er å forvente for en bacheloroppgave.

En helt sentral grunn til at resultatene fra prosjektarbeidet kom i mål, er det faktum at gruppen til enhver tid var innstilt på å raskt endre fokus og tilnærming til problemstillingen. Dersom det viste seg at metoden som ble benyttet var gruppen villig til å ta et steg tilbake, for å skape mer skalerbare og hensiktsmessige løsninger på sikt. Et eksempel på dette er hvordan støyreduksjon ble forsøkt implementert, men at det senere viste seg å ikke gi tilfredsstillende resultater gitt begrensningene i dataen som ble samlet inn. Et annet eksempel på dette var hvordan gruppen ble nødt til å endre og formalisere testprosessen etter hvert som nye behov dukket opp i maskinlæringsmetodene. Et annet poeng gruppen har erfart opp igjennom studiene, er nytten av å jobbe på ulike teknologier og arbeidsoppgaver parallelt. For det første unngår man i stor grad eventuelle flaskehalser som ofte oppstår i løsninger av vanskelige problemstillinger. For det andre gir dette muligheter til å bruke erfaringer fra et domene inn i et annet. Dersom gruppen eksempelvis hadde ventet med evaluering av øyedata til etter designet av scenene, hadde gruppen potensielt ikke designet eksperimenter som samlet inn data som ikke var egnet for videre bruk i maskinlæring.

Det største læringsutbyttet fra prosessen er i gruppens øyne erfaringen med å jobbe med hele forskningsprosessen, og alle dens aspekter. Gruppemedlemmene har eksempelvis aldri jobbet med design av eksperimenter, eller datainnsamling. Å få erfart den tidkrevende og iterative prosessen som går inn i å designe, samle inn og behandle data har gitt uvurderlig ingeniørfaglig innsikt. Gruppen har også satt seg inn i et domene som ved oppstartsfasen var helt ukjent, samt å sette seg inn i en tidligere bacheloroppgave. Gruppen var forberedt på at dette kom til å være utfordrende, men ser i ettertid at dette var en enda mer tidkrevende prosess enn først antatt. Samtidig har dette resultert i et prosjekt hvor gruppen i stor grad har kontrollert selv hvilken retning man skulle ta forskningen, samtidig som at en hadde verktøy og innsikt til å tilpasse alle deler av prosjektet etter behov. Det å kunne designe og samle inn dataen selv, har gjort at en kunne utforske domenet uten å være avhengig av et ferdigstilt datasett. På denne måten har gruppens kunnskap til prosjektarbeidet bredere enn det tidligere har vært, samtidig som man har hatt mulighet til å fordype seg i både det teoretiske og den praktiske anvendelsen av øyedata.

6 Konklusjon

Resultatene som er presentert fra dette prosjektet viser et svært lovende potensial for forskning på øyebevegelser ved hjelp av eyetracking i virtuell virkelighet. Samtidig har anvendelsen av denne øyedataen gitt ny innsikt i hvordan en kan benytte seg av maskinlæring for å klassifisere, identifikasjon og prediksjon av øyets bevegelser. Eksperimentdesign, samt datainnsamling, evaluering og visualisering har vist at implementasjonene av algoritmene gir svært varierte klassifiseringer av øyets bevegelser. Samtidig kan en identifisere store forskjeller basert på brukerens visuelle input, gitt at eksperimentene design og gjennomførelse begrenser støy i form av hodebevegelser. Denne

innsikten har videre avdekket at blikkets posisjon over tid som en egnet metrikk for å klassifisere visuell input og skille på individuell adferd i et VR-miljø.

Det konvolusjonelle nevrale nettverket viste at det var mulig å skille på scener ved å visualisere blikkets posisjon over tid. Det beste resultatet på uavhengig testsett kom på 63.35 prosent ved bruk av et intervall på tre sekunder. På helt usett data oppnådde LSTM-modellen initiell presisjon på 74.10 prosent på alle fire scenen, som ble økt til 89.28 prosent når many-ball-still ble fjernet fra datasettet. Forvirringsmatrisen fra både CNN og LSTM viste at many-ball-still og many-ball-disappear var for like til å skilles med høy presisjon, gitt det benyttet datagrunnlaget. LSTM klarte videre å skille gruppens tre medlemmer med en prosent på 67.59 på tvers av alle fire scenene. Ved å spesifisere trening og testing til kun many-ball-disappear ble presisjonen 92.75 prosent. Resultatene viser at det på bakgrunn av øyebevegesler, er mulig å identifisere hvem som bruker HMD-et med svært høy presisjon. For å videre utforske spesifikke forskjeller i brukeres fikseringer, kan scenene one-ball-still og one-ball-move benyttes. Her foreslår gruppen et større datagrunnlag.

Ved å benytte oss av forecasting med en N-BEATS modell klarte vi å bevise at prediksjon av sakkader med høy treffsikkerhet er mulig. Når modellen predikerte på et uavhengig testsett med usett data fikk vi en root mean square error verdi på 0.044. Datasettet vårt er skalert, og verdiene i settet befinner seg mellom 0 og 1. En RMSE verdi på 0.044 i et datasett med verdier mellom 0 og 1 er et svært godt resultat, og viser at modellen vår er god på å komme med numeriske tilnærminger på de fremtidige verdiene til blikkets vektorer. Dog er modellen avhengig av et klassifisert datasett, noe tidligere diskutert kan være vanskelig å vite med sikkerhet om man har riktig eller ikke. Uavhengig av det så vier resultatene av forecasting modellen er godt egnet til å tilnærme og lære seg trender i et datasett bestående av blikkdata, for å så predikere kommende verdier.

Hensikten med dette prosjektarbeidet var å evaluere tidligere arbeid, og å benytte denne for å gjøre videre forskning på området. Selv om enkelte deler av oppgaven har blitt revidert i løpet av prosjektarbeidet, er essensen bevart. Den skriftlige rapporten sammen med de implementerte verktøyene gir 3D Motion Technologies et utgangspunkt for videre forskning innenfor domenet som er eyetracking.

Referanser

[1] Artificial nueral network tutorial. URL: https://www.javatpoint.com/artificial-neural-network.

[2] Common loss functions in machine learning. URL: https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23.

[3] Ml: Underfitting and overfitting. URL: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/.

[4] An overview of machine learning optimization techniques. URL: https://serokell.io/blog/ml-optimization.

[5] D. S. Zee A. G. Lasker. Ocular motor abnormalities in huntington's disease, 2022. Hentet: 19.05.2022. URL: https://pubmed.ncbi.nlm.nih.gov/9425536/.

[6] MD A. Nicole Somani, Bayan Al Othman. Saccade, 2022. Hentet: 13.05.2022. URL: https://eyewiki.aao.org/Saccade.

[7] Akash Agnihotri. Implementing real-time object detection system using pytorch and opencv, 2021. Hentet: 13.05.2022. URL: https://towardsdatascience.com/implementing-real-time-object-detection-system-using-pytorch-and-opencv-70bac41148f7.

[8] D2I AI. Recurrent nueral networks. Hentet: 01.02.2022. URL: http://d2l.ai/chapter_recurrent-neural-networks/rnn.html.

[9] Amazon. Model fit: Underfitting vs overfitting. URL: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html.

[10] Meraldo Antonio. Word embedding, character embedding and contextual embedding in bidaf — an illustrated guide. URL: https://towardsdatascience.com/the-definitive-guide-to-bidaf-part-2-word-embedding-character-embedding-and-contextual-c151fc4f05bb.

[11] Nir Arbel. URL: https://medium.datadriveninvestor.com/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577.

[12] G.R. Barnes. Cognitive processes involved in smooth pursuit eye movements. *Brain and Cognition*, 68(3):309–326, 2008. A Hundred Years of Eye Movement Research in Psychiatry. URL: https://www.sciencedirect.com/science/article/pii/S0278262608002650, doi:https://doi.org/10.1016/j.bandc.2008.08.020.

[13] Nicolas Chapados Yoshua Bengio Boris N. Oreshkin, Dmitri Carpov. N-beats: Neural basis expansion analysis for interpretable time series forecasting, 2022. Hentet: 28.04.2022. URL: https://arxiv.org/abs/1905.10437.

[14] Viviane Clay, Peter König, and Sabine Koenig. Eye tracking in virtual reality. *Journal of Eye Movement Research*, 12, 04 2019. Hentet: 13.05.2022. doi:10.16910/jemr.12.1.3.

[15] Darts, 2020. Hentet: 14.05.2022. URL: https://unit8co.github.io/darts/generated_api/darts.models.forecasting.html.

[16] Andrew Duchowski. Eye tracking methodology - theory and practice. URL: https://link.springer.com/book/10.1007/978-1-84628-609-4.

[17] SR Research EyeLink. URL: https://www.sr-research.com/eye-tracking-blog/background/eye-tracking-terminology-eye-movements/.

[18] OpenCV for Unity. How to use pytorch models in unity, 2022. Hentet: 4.04.2022. URL: https://enoxsoftware.com/opencvforunity/.

[19] A. Graves. Supervised sequence labelling with recurrent neural networks. studies in computational intelligence. Hentet: 10:03:2022. URL: https://www.cs.toronto.edu/~graves/preprint.pdf.

[20] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. neural networks, 18(5-6):602–610. Hentet: 10:03:2022. URL: https://pubmed.ncbi.nlm.nih.gov/16112549/.

[21] HårekHaugerudI. sanntid (it). Hentet: 19.05.2022. URL: https://snl.no/sanntid_-_IT.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. Hentet: 13.05.2022. doi:10.1162/neco.1997.9.8.1735.

[23] Kenneth Holmqvist, Marcus Nyström, and Fiona Mulvey. Eye tracker data quality: What it is and how to measure it. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, page 45–52, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2168556.2168563.

[24] IBM. What is computer vision. Hentet: 01.03.2022. URL: https://www.ibm.com/se-en/topics/computer-vision.

[25] Julien I.E.Hoffman. Basic biostatistics for medical and biomedical practitioners (second edition), 2019. URL: https://www.sciencedirect.com/topics/medicine-and-dentistry/friedman-test.

[26] FRITZ LABS INCORPORATED. Object detection guide. Hentet: 19.05.2022. URL: https://www.fritz.ai/object-detection/.

[27] Influxdata. What is time series data? Hentet: 19.05.2022. URL: https://www.influxdata.com/what-is-time-series-data/.

[28] Influxdata. Time series forecasting methods, 2022. Hentet: 22.03.2022. URL: https://www.influxdata.com/time-series-forecasting-methods/.

[29] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. URL: https://arxiv.org/abs/1506.02025?context=cs.

[30] Jan K. S. Jansen, 2022. Hentet: 13.05.2022. URL: https://sml.snl.no/vestibulo-okul%C3%A6r_refleks.

[31] Ajitesh Kumar. Why use random seed in machine learning? Hentet: 19.05.2022. URL: https://vitalflux.com/why-use-random-seed-in-machine-learning/.

[32] Great Learning. Multinomial naive bayes explained. Hentet: 01.04.2022. URL: https://www.mygreatlearning.com/blog/what-is-machine-learning.

[33] Dillon J. Lohr, Lee Friedman, and Oleg V. Komogortsev. Evaluating the data quality of eye tracking signals from a virtual reality system: Case study using smi's eye-tracking HTC vive. *CoRR*, abs/1912.02083, 2019. URL: http://arxiv.org/abs/1912.02083, arXiv:1912.02083.

[34] Nergiz Ercil Menekse Dalveren, Gonca Gokce; Cagiltay. What eye tracking teaches us about advertising and consumerbehavior, 2017. URL: https://www.alistdaily.com/digital/what-eye-tracking-teaches-us-about-advertising-and-consumer-behavior.

[35] Nergiz Ercil Menekse Dalveren, Gonca Gokce; Cagiltay. Evaluation of ten open-source eye-movement classification algorithms in simulated surgical scenarios, 2019. URL: http://hdl.handle.net/11250/2636329.

[36] Ramberg Moan, Nygaard. Eyetracking i vr, 2022. Hentet: 17.05.2022. URL: https://gitlab.stud.idi.ntnu.no/bachelor102/unity-vr-bachelor/-/wikis/uploads/b9abb27e9129ce7be7bcdd4ea5bd0e13/Bachelor_-_Tidligere_oppgave_-_Rapport.pdf.

[37] Ramberg Moan, Nygaard. Forstudierapport, 2022. Hentet: 17.05.2022. URL: https://gitlab.stud.idi.ntnu.no/bachelor102/unity-vr-bachelor/-/wikis/uploads/465f203e181a72e2374567235474465c/forstudierapport.pdf.

[38] Nvidia. Accelerated computing - training, 2022. Hentet: 03.05.2022. URL: https://developer.nvidia.com/accelerated-computing-training.

[39] Marcus Nyström and Kenneth Holmqvist. An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. URL: https://doi.org/10.3758/BRM.42.1.188.

[40] The University of Chicago. 14.2 types of eye movements, 2017. Hentet: 01.03.2022. URL: https://www.youtube.com/watch?v=eb3h7oUqcTI.

[41] Anneli Olsen. The tobii i-vt fixation filter. URL: https://www.tobiipro.com/siteassets/tobii-pro/learn-and-support/analyze/how-do-we-classify-eye-movements/tobii-pro-i-vt-fixation-filter.pdf.

[42] Medium. H. Pedamallu. Rnn vs gru vs lstm. Hentet: 01.02.2022. URL: https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573.

[43] André Pereira. How to use pytorch models in unity, 2020. Hentet: 10.05.2022. URL: https://medium.com/@a.abelhopereira/how-to-use-pytorch-models-in-unity-aa1e964d3374.

[44] M. Phi. Hentet: 01.02.2022.

[45] Tobii Pro. The tobii pro fixation filters. URL: https://tobii.23video.com/the-tobii-pro-fixation-filters-eye-movement.

[46] Pytorch. Lstm. Hentet: 10.11.2021. URL: https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html.

[47] K. Rayner. Eye movements in reading and information processing: 20 years of research. URL: https://pubmed.ncbi.nlm.nih.gov/9849112/.

[48] L. Østby S. Jensen, S. Mogen. Møtereferat alsam, 2022. Hentet: 15.05.2022. URL: https://gitlab.stud.idi.ntnu.no/bachelor102/unity-vr-bachelor/-/wikis/uploads/a7f79116225e72bef56aa0acb18bd4bb/Main_report.pdf.

[49] Unit8 SA. Darts quickstart, 2022. Hentet: 01.05.2022. URL: https://developer-express.vive.com/resources/vive-sense/eye-and-facial-tracking-sdk/overview/.

[50] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. hentet: 01.03.2022. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[51] Dario D. Salvucci and Joseph H. Goldberg. Identifying fixations and saccades in eye-tracking protocols. In *Proceedings of the 2000 Symposium on Eye Tracking Research amp; Applications*, ETRA '00, page 71–78, New York, NY, USA, 2000. Association for Computing Machinery. doi:10.1145/355017.355028.

[52] Shane D. Sims, Vanessa Putnam, and Cristina Conati. Predicting confusion from eye-tracking data with recurrent neural networks. *CoRR*, abs/1906.11211, 2019. Hentet: 08.05.2022. URL: http://arxiv.org/abs/1906.11211, arXiv:1906.11211.

[53] Rajeev Ranjan Sreekanth Menom, 2022. Hentet: 01.05.2022. URL: https://www.genpact.com/insight/technical-paper/the-evolution-of-forecasting-techniques-traditional-versus-machine-learning-methods.

[54] Statistics How To. Mean absolute percentage error (mape). Hentet: 17.05.2022. URL: https://www.statisticshowto.com/mean-absolute-percentage-error-mape/.

[55] Statistics How To. Mean squared error definition. Hentet: 17.05.2022. URL: https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/mean-squared-error/.

[56] Tobii. Vive pro eye with tobii, 2022. Hentet: 02.04.2022. URL: https://vr.tobii.com/integrations/htc-vive-pro-eye/.

[57] Unity, 2022. Hentet: 02.05.2022. URL: https://docs.unity3d.com/Packages/com.unity.barracuda@0.2/manual/index.html.

[58] Valve. Steamworks-dokumentasjon, 2022. Hentet: 02.04.2022. URL: https://partner.steamgames.com/doc/features/steamvr/info.

[59] Nicolas Vandeput, 2019. Hentet: 16.05.2022. URL: https://towardsdatascience.com/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d.

[60] Zhe wu and Panagiotis Christofides. Economic machine-learning-based predictive control of nonlinear systems. *Mathematics*, 7:494, 06 2019. doi:10.3390/math7060494.

[61] Yuehan Yin, Chunghao Juan, Joyram Chakraborty, and Michael P. McGuire. Classification of eye tracking data using a convolutional neural network. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 530–535, 2018. doi:10.1109/ICMLA.2018.00085.

[62] J.M. Zheng, K.W. Chan, and I. Gibson. Virtual reality. *IEEE Potentials*, 17(2):20–23, 1998. doi:10.1109/45.666641.

[63] L. Østby. Gammel gruppe møtereferat, 2022. Hentet: 17.05.2022. URL: https://gitlab.stud.idi.ntnu.no/bachelor102/unity-vr-bachelor/-/wikis/uploads/a0ffd0454f986fc45439784780daf8eb/M%C3%B8tereferat_gammel_gruppe_17.03.docx.

[64] L. Østby. Møtereferat alsam, 2022. Hentet: 16.05.2022. URL: https://gitlab.stud.idi.ntnu.no/bachelor102/unity-vr-bachelor/-/wikis/uploads/b112f332a552e5e4b766a484321c5b53/M%C3%B8tereferat_06.04.2022.docxf.

Vedlegg

Vision Document

102

IDATT2900-102
Visjonsdokument

Versjon 1.3

Vedlegg

Vision Document

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
14.01.2022	1.0	Førsteutkast	Simon Jensen Stian Mogen Lars Brodin Østby
19.01.2022	1.1	Revidert etter oppstartsmøte med veileder/oppdragsgiver, og lest forprosjektplan og rapport	Simon Jensen Stian Mogen Lars Brodin Østby Alexander Holt
24.01.2022	1.2	Oppdatert struktur og kildehenvisning	Stian Mogen

Vedlegg

Vision Document

Innholdsfortegnelse

Vedlegg

Vision Document

1. Innledning

Dette dokumentet har som hensikt å beskrive de overordnede kravene og målene satt for avsluttende bacheloroppgave IDATT2900-120 for dataingeniør studieprogrammet, gruppe 102. Bacheloroppgaven skrives vårsemesteret 2022 fra perioden januar til mai. Prosjektforløpet er en videreutvikling av bacheloroppgaven skrevet av Martin Moan, Marius Nygård og Håvard Ramberg, og vil dermed i stor grad basere seg på arbeidet gjort i dette prosjektet. Disse konklusjonene og betraktningene vil vurderes og forsøkt forbedres.

Hensikten med oppgaven er å evaluere og videreutvikle et eksisterende prosjekt utført i sammenheng med VR teknologi, med tanke på prediksjon av øyebevegelse samt kognitivt fokus/oppmerksomhet. Tidligere prosjekt var i allere høyeste grad forskningsbasert. Denne kunnskapen skal gi innsikt for utvikling av brukbare systemer i dette prosjekter, samt ny forskning og kunnskap om fagfeltet.

2. Sammendrag problem og produkt

2.1 Problemsammendrag

Vi vil nå videreføre denne oppgaven, og med utgangspunkt i det tidligere arbeidet gjort, er vi interessert i å samle inn mer data for å kunne utføre en grundigere evaluering av det tidligere arbeidet.

I tillegg til dette er vi interessert se på brukbarheten av systemet med tanke på to nye aspekter: prediksjon av øyebevegelser (gitt øyets nylige bevegelser, kan vi forutsi med rimelig sannsynlighet hvor det kommer til å bevege seg et kort stykke inn i fremtiden?), og kognitiv oppmerksomhet/fokus (hva en person faktisk er oppmerksom på i det virtuelle miljøet).

Det tidligere arbeidet på prosjektet gir god innsikt og nyttig data for å kunne predikere oppmerksomhet i VR, dette skal videreutvikles (M. Moan, M. Nygård, H. Ramberg, 2020). For å bedre forstå hvordan et menneske reagerer i ulike situasjoner må ens blikk oppførsel undersøkes og analyseres. En dyp analyse av øyebevegelse kan hjelpe forskere å forstå hvordan øyet beveger seg over synsfeltet, og hva den tar fokus på under ulike situasjoner. Den innsamlede dataen kan bli brukt i videre forskning eller det kan benyttes i utvikling av ny teknologi, enten som en utgangspunkt eller som et ledd i et større produkt.

Problem med	Datagrunnlag og maskinlæringsalgoritmer som ikke på en tilfredsstillende måte egner seg til sanntidsprediksjon av oppmerksomhet
berører	3D-Motion-Technologies
som resultatet av dette	Blir det umulig å med høy presisjon registrere hva som er sakkader og hva som er frivillig øyebevegelser med konkret fokus.
en vellykket løsning vil	På forminske støy og predikere fokus med høy nøyaktighet.

Problem med	Tilgjengelighet av opensource prosjekter og innsamling av data vedrørende eyetracking
berører	Utviklere av større systemer innen dette fagområdet

Vedlegg

Vision Document

som resultatet av dette	Forsinkes eller direkte blokkere utvikling og forsking på ny og annen teknologi som bygger på, eller er avhengig av, resultatene og dataen samlet og analysert i et slikt prosjekt.
en vellykket løsning vil	Vil resultatene fra prosjektet bli brukt til videre forskning i fremtiden etter ferdigstilling.

2.2 Produktsammendrag

For	3D Motions Technologies, samt uavhengige forskere i fremtiden
som	Har et behov for å tolke øyets fokus og bevegelsesmønster i et VR-miljø
produktet navngitt	102
som	I sanntid predikerer fokus og bevegelse basert på innlest data fra øyet i et VR-miljø, ved bruk av sensorer.
I motsetning til	Det opprinnelige prosjektet
Har vårt produkt	Tilstrekkelig filtrert innlest data til trening med lite støy, som sammen med forbedrede maskinlæringsalgoritmer gir høy grad av nøyaktighet.

3. Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen
Alexander Holt	Stakeholder 3D Motion Technologies	Bistår arbeidet med innspill, prioriteringer og godkjenning av krav til prosjektet.
Alexander Holt	Universitetslektor ved Institutt for datateknologi og informatikk, NTNU	Gir tilbakemelding og veiledning gjennom prosjektarbeidet.
Institutt for datateknologi og Informatikk	Institutt tilhørende NTNU, som studentene er en del av.	Stiller de akademiske formelle retningslinjene og kravene til bacheloremnet IDAT2900

3.2 Oppsummmering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av
Teamet	Utvikler av systemet	Tilegner seg kunnskap og anvender denne i utviklingen av systemet og forskningen	Simon Jensen, Stian Mogen og Lars Brodin Østby
Forsøksperson	Spiller scenene i VR-miljø og får blikket registrert	Tester programmet og brukes til bidrar med treningsdata	Frivillige

Vedlegg

Vision Document

Forskere og teknologer	Bruker maskinlæring til å registrere innsamlet data og predikerer basert på denne	Utvikler og tilbakemeldinger	Studenter og instituttet
Medisinsk personell	Bruker dataen til videre forskning på øyet	Eventuelle tilbakemeldinger	N/A

3.3 Brukermiljøet

Prosjektet er forskningsbasert, og resultatet vil i stor grad være todelt. VR-miljøet, hvor innhentingen av data vil skje utvikles og testes ut ved forskningslab på NTNU. Denne prosessen kan også i fremtiden bli gjort fra enkeltpersoners hjem gitt tilstrekkelig utstyr, derimot er det i all hovedsak snakk om et produkt som brukes i et kontrollert testmiljø på lab, eventuelt av medisinsk personell ved sine institutter.

Selve produktet skal fungere i et VR-miljø utviklet med spillmotoren Unity. Denne utvikles i C#.

Den andre delen vil være resultatene fra forskningen som brukes. Rapporten skal i seg selv gi nyttig innsikt og data som fagfolk kan bruke i egne forskningsprosjekter. Denne delen vil være uavhengig fra labben, og kunne bli benyttet hvor som helst. Det er derfor nødvendig at alt som produserer dokumenteres nøye og godt.

3.4 Sammendrag av brukernes behov

Behov	Prioritet	Påvikrer	Dagens løsning	Foreslått løsning
Samle brukbar blikkdata med sensorer	Kritisk	Datainnsamling	Implementer i prosjektet fra før med mye stør	Forbedre etter behov, blant annet støy
Filtrere og bearbeide data for bruk i maskinlæring	Høy	Utviklere av maskinlæring smodell	Implementert i prosjektet	Funksjonalitet for bearbeiding og filtrering av data før trening
Automatisering av klassifisering	Lav	Maskinlæring smodellen	Manuell klassifisering	Automatisere klassifiseringen ved innhenting av data
Tolkning av øyebevegelse	Høy	Maskinlæring smodell		
VR-miljø med spill	Kritisk	Brukere og forskningsper soner	Implemetert i prosjektet fra før	Forbedre etter behov, utvikle flere scener
Sanntids tilbakemelding på blikk	Lav	Brukere og utviklere	Ingen sanntids tilbakemelding	Fargekodet på blikkets posisjon i VR-miljø

3.5 Alternativer til vårt produkt

Dagens åpenbare alternativ blir det aktuelle prosjektet som vi skal videreutvikle. Forskjellene og forbedringene fra dette produktet står beskrevet tidligere i dokumentet.

Vedlegg

Vision Document

4. Produktoversikt

4.1 Produktets rolle i brukermiljøet

Produktet skal kunne brukes til innsamling av data, for bruk til å trene opp maskinlæringsmodellen. Modellen som predikerer øyebevegelse samt dataen som innsamles skal kunne brukes på testere, og potensielt av medisinsk personell i konsultasjon.

Utviklingsprosjektet skal ende i et produkt som er intuitivt og brukervennlig, dog vil hovedfokuset ligge på funksjonalitet. Produktet sin hensikt vil være å vise forskningen og resultatene, og vil i mindre grad være en kommersiell programvare som skal distribueres.

4.2 Forutsetninger og avhengigheter

- Prosjektets fremgang vil være avhengig av kvaliteten på det tidligere prosjektet som skal jobbes videre på, og det prosjektet sin dokumentasjon. I hvilken grad dette arbeidet er skalerbart og lar seg forbedres.
- Mulighet for å være på labb på universitet for å kunne benytte seg av VR brukermiljøet.
- Tilgang til nok data til trening av modell
- Lavt nok smittetrykk til å få inn nok testpersoner
- Brukbarheten til HTC Vive Pro Eye sammen med Unity VR-miljø for tilfredsstillende innsamling av data

5. Produktets funksjonelle egenskaper

- Maskinlæringsmodell som med svært høy presisjon kan predikerer kognitivt fokus basert på sakkadene til øyet
- Maskinlæringsmodell som med noe presisjon kan predikere øyets fremtidige bevegelse basert på tidligere bevegelse
- VR-miljø som lar brukere utforske
- Sensorer som plukker opp øyets bevegelse og blikk
- Skille på når blikket er fokus eller sakkader
- Kalibrering av blikk slik at tester er med likt utgangspunkt og reproduserbare (K. Harezlak, P. Kasprowski, 2018)
- Lagring av dataen
- Et enkelt brukergrensesnitt for databehandling

6. Ikke-funksjonelle egenskaper og andre krav

- Høy nøyaktighet for prediksjon av fokus
- Høy nøyaktighet for prediksjon av øyebevegelse
- Nøyaktig deteksjon av blikkfanget i VR-miljøet
- API-et skal være godt dokumentert, både i koden og på prosjektets wiki side.
- API-et skal benytte seg av C# for Unity VR-miljø
- Programmet skal kunne brukes med HTC Vive Pro
- Programvaren skal være pålitelig og ha god ytelse.
- Programmet skal følge WCAG sine prinsipper om begripelighet er nødvendig.

7. Referanser

1. M. Moan, M. Nygård, H. Ramberg (2020), *Eyetracking i VR,* 3D Motion Technologies
2. K. Harezlak, P. Kasprowski (2018) *Application of eye tracking in medicine: A survey, research issues and challenges.* Hentet fra: www.sciencedirect.com/science/article/pii/S0895611117300435

Vedlegg

Forprosjektplan

Prosjektnr

Oppgave 102
Forprosjektplan

Versjon 1.0

Vedlegg

Forprosjektplan

Prosjektnr

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
18.01.2022	1.0	Førsteutkast	Simon Jensen Stian Mogen Lars Brodin Østby

Vedlegg

Forprosjektplan

Prosjektnr

Innholdsfortegnelse

Vedlegg

Forprosjektplan

Prosjektnr

1. Mål og rammer

1.1 Orientering.

Oppgaven ble publisert sammen med andre aktuelle oppgaver i emnet på Blackboard. Utgiver av oppgaven var 3D Motion Technologies, med veileder Alexander Holt. Etter en lengre prosess falt gruppens prioriteringer på tre aktuelle oppgaver, hvor denne var det foretrukne alternativet. Gruppen har tidligere jobbet med Holt og 3D Motion Technologies, og har gode erfaringer fra prosjektarbeidet både arbeidsmessig og resultatmessig. Selve oppgaven var spesielt interessant, da det ga gruppen en mulighet til å jobbe med maskinlæring, som er et gjennomgående interesseområde i hele gruppen. Samtidig er utvikling i VR-noe gruppen ikke har tidligere erfaringer med, men også et stort ønske om å jobbe med. Ettersom vi tidligere kun har utvikler prosjekter fra grunnen av vil det også nå være interessant å bygge videre på tidligere arbeid. Gruppen har tro på at dette kan innsikt og erfaringer om gruppeprosjekter fra en ny side.

1.2 Problemstilling / prosjektbeskrivelse

Det har tidligere vært gjennomført en bacheloroppgave hvor det ble implementert forskjellige VR-miljø/eksperimentscener for å kunne forske på å bruke maskinlæring for å hurtig kunne avgjøre en persons blikk-fokus i VR. Vi vil nå videreføre denne oppgaven, og med utgangspunkt i det tidligere arbeidet gjort, er vi interessert i å samle inn mer data for å kunne utføre en grundigere evaluering av det tidligere arbeidet.

I tillegg til dette er vi interessert se på brukbarheten av systemet med tanke på to nye aspekter: prediksjon av øyebevegelser (gitt øyets nylige bevegelser, kan vi forutsi med rimelig sannsynlighet hvor det kommer til å bevege seg et kort stykke inn i fremtiden?), og kognitiv oppmerksomhet/fokus (hva en person faktisk er oppmerksom på i det virtuelle miljøet).

Oppgaven er av akademisk art og må sies å være forskningsrettet, men det vil likevel være nødvendig å kunne implementere ny funksjonalitet på det eksisterende systemet. Det er et mål at resultatet av oppgaven blir brukt i videre forskning, og vi har andre parallelle prosjekter som vil kunne dra nytte direkte nytt av arbeidet som vil bli gjort.

1.3 Resultatmål

- Hvert gruppemedlem skal jobbe 500 timer

Hvert gruppemedlem har 500 timer (pluss minus 10%) til disposisjon. Dette betyr at det forventes at gruppen jobber: ikke mindre enn 1350 timer, ikke mer enn 1650 timer.

- Prosjektet innleveres etter rammene definert i visjonsdokumtentet.

Vedlegg

Forprosjektplan

Prosjektnr

Prosjektets konkrete målsetninger defineres i visjonsdokumentet. Gruppen skal innenfor timene definert i punkt 1, innfri kravene satt til "Høy" eller større viktighet.

- Innsamling av brukbar data for øyet i VR-miljø

Vi skal samle inn treingsdata gjennom bruk av VR-miljøet. Denne skal være av god nok kvalitet til å kunne benyttes til trening av maskinlæringsmodeller.

- Trene maskinlæringsmodeller fra innsamlet data

I prosjektet skal det trenes maskinlæringsmodeller for å bestemme kognitivt fokus. Det skal også trenes opp modell for å predikere fremtidig bevegelse basert på tidligere posisjon. Denne skal kunne predikere med høyere sikkerhet enn vilkårlig gjetting.

- Automatisk evaluering av modeller og algoritmer

I tidligere prosjekt ble resultatene i stor grad kvalitetssikret manuelt. Vi skal implementere en mer skalerbar løsning for evalueringen.

1.4 Effektmål

- Forske på bruk av eyetracking i VR-miljø

Prosjektet skal gjennom forløpet gi innsikt i hvorvidt en kan benytte seg av et VR-miljø for forskning på øyets bevegelse, og detektere dette i sanntid. Om dette lar seg gjøre er det mulig å benytte forskningen til utvikling av verktøy i ulike næringer. Eksempelvis kan forskningen bli avgjørende for å finne ut hvor bilister har blikkfokus ved kjøreopplæring i VR.

- Forske på bruk maskinlæring til å avgjøre kognitivt fokus

Prosjektet skal gi økt innsikt i hvor stor grad man kan benytte maskinlæring til å predikere kognitivt fokus basert på øyets bevegelse. Svarene fra denne prosessen kan gi nyttig innsikt i videre benyttelse av maskinlæring for prediksjon av fokus og øyet generelt.

1.5 Prosessmål

- Utvikling av kompetanse for gruppens studenter og aktører

Vedlegg

Forprosjektplan

Prosjektnr

Et slikt prosjekt vil ikke bare være nyttig basert på prosjektets resultater, men vil også gi god erfaring for de aktuelle studentene som jobber med prosjektet, samt oppdragsgiver fra instituttet. Prosjektets gang vil gi innsikt i hvordan et slikt forskningsprosjekt gjennomføres på en hensiktsmessig måte, og i hvor stor grad det er gjennomførbart gitt tilgjengelige ressurser for gruppen.

1.6 Rammer

Prosjektet vil bli utført i perioden 14.01.2022 - 27.05.2022. Tiden vil bli delt opp i perioder med oppstart, prosjekt utvikling parallelt med datainnsamling- og behandling, forbedring/forskning av teknologien og avslutter med dokumentasjon og presentasjon av prosjektet.

Oppgaven bygger videre på en tidligere bachelor prosjekt, som er et oppsatt VR brukermiljø. Det vil være nødvendig for gruppen å ha tilgang til utstyret og oppsettet som benyttes i kjøringen av dette tidligere prosjektet. Det vil også være hensiktsmessig å ha tilgang til kraftig maskinvare for effektiv videreutvikling av maskinlæringsmodeller.

2. Organisering

3D Motion Technologies: NTNU basert teknologiforetak som spesialiserer seg på blant annet bevegelsesbasert velferdsteknologi. Oppdragsgiver av oppgaven på vegne av NTNU, representert av Alexander Holt.

Institutt for Datateknologi og Informatikk: Instituttet studieretningen faller under, og tilhører NTNU. Instituttet stiller med veileder Alexander Holt.

3. Gjennomføring

3.1. Hovedaktiviteter.

Prosjektarbeidet blir bestående både av generell utforskning av problemstillingen og eventuelle løsninger, men også konkrete definerte arbeidsoppgaver som legger til rette for forskningen. Disse kan defineres på følgende måte. Dette blir det første som gjøres i prosjektarbeidet.

- Sette seg inn i tidligere oppgave

Alle gruppens medlemmer er avhengig av å sette seg inn i det teoretiske arbeidet som er blitt gjort på prosjektet fra før. Dette innebærer å lese forprosjektstudiet, hovedrapporten, samt få en oversikt over annen relevant litteratur rundt emnet.

- Bli kjent med utstyr og programvare på Vizlab.

Vedlegg

Forprosjektplan

Prosjektnr

For å kunne utvikle egne løsninger trenger gruppen å være kjent med utviklingsmiljøet, samt det fysiske utstyrets styrker og begrensninger. Denne jobben begynner i uke 4. Her er gruppen avhengig av tilstrekkelig tilgang på utstyr.

- Samle inn data

Prosjektarbeidet avhenger av et godt datagrunnlag. Derfor blir det nødvendig å utvikle flere testmiljøer for datainnsamling og klassifisering. Dette skal videre brukes til å samle inn brukbar data fra testpersoner. Utvikling og utforskning av datainnsamling begynner i uke 5. Dette arbeidet kan potensielt bli påvirket av restriksjoner som følge av pandemien.

- Utforskning av algoritmer og maskinlæringsmodeller

Denne aktiviteten er avhengig av datagrunnlaget fra punktet over. Her vil ulike metoder for sanntidsklassifisering av data bli utforsket, slik at gode resultater kan samles inn til analyse og diskusjon i sluttrapporten. Denne delen av prosjektet starter forhåpentligvis i månedsskifte februar/mars. Utforskningen vil basere seg på dataen som gruppen samler inn fra testpersoner.

- Rapportskriving

Et kontinuerlig arbeid alle gruppens medlemmer er ansvarlig for. Denne vil bli skrevet kontinuerlig gjennom prosjektarbeidet, og avsluttes ukene før innlevering.

Disse aktivitetene gir utgangspunktet for det foreløpige utkastet for fremdriftsplanene definert i Ganttdiagram vedlegg 6.1.

3.2. Milepæler.

Obligatoriske arbeidskrav:

- 28.01.2022 - Forprosjektplan
- 28.03.2022 - Poster/plakat enten avslutningsvis eller underveis i prosessen
- Uke 13 (28.03 - 03.04.2022) - Muntlig presentasjon på engelsk underveis i prosessen
- 20.05.2022 - Dokumentasjon av arbeidsprosess, inklusiv timeliste, statusrapporter, møteinnkalling og referat
- 27.05.2022 - Muntlig fremføring av den innleverte bachelor oppgaven

Disse datoene er de formelle obligatoriske datoene listet nevnt i emnebeskrivelsen. Gruppenes egne frister og foreløpige fremdriftsplan er vedlagt i Gantt diagrammet i 6.1. Det er verdt å merke seg at disse fristene er et foreløpig utkast, og at som følge av prosjektets forskningspreg vil faste frister være vanskelig å sette i forkant. Gruppen forholder seg derfor til dette som et utgangspunkt for prosjektforløpet.

Vedlegg

Forprosjektplan

Prosjektnr

4. Oppfølging og kvalitetssikring

4.1 Kvalitetssikring.

Kvalitetssikring på et forskningsprosjekt av denne art er svært viktig. Først og fremst vil dette skje via jevnlige tilbakemeldinger fra både veileder og oppdragsgiver. Samtidig er det ønskelig å få inn testpersoner som kan bruke produktet, og gi sine erfaringer og tilbakemeldinger. Avslutningsvis kan det være hensiktsmessig med tilbakemeldinger fra helsepersonell.

4.2 Rapportering.

Slutten av prosjektarbeidet vil ende i en hovedrapport, som blant annet skal beskrive hele prosessen gjennom arbeidet, sammen med produktet som leveres.

Underveis så rapporteres fremdriften til Alexander Holt, som fungerer som både veileder og oppdragsgiver. Veiledning skjer med utgangspunkt i møter annen hver uke. Dette kan økes etter behov. Som oppdragsgiver vil tilbakemeldinger og spørsmål komme fortløpende.

5. Risikovurdering

Hendelse	Sannsynlighet	Konsekvens	Tiltak
Mangel på data til maskinlæringsmodell	Høy	Det blir vanskelig å danne gode, generelle modeller	Prioritere innsamling av data tidlig i prosjektet
Uklare mål og rutiner	Middels	Prosjektet blir dårlig disponert og eventuelle mangler kan oppstå	Dokumentere og planlegge jevnlig
Dårlige prioriteringer som følge av stor oppgave	Middels	Viktige områder blir tilsidesatt og eventuelle mangler kan oppstå	Avtale hovedprioriteringer med oppdragsgiver og eventuelle utvidelser
Dårlig oppfølging	Lav	Fører til dårlige prioriteringer og gruppen kan risikere å gjøre gale antagelser	Søke jevnlig, motiverende kontakt med veileder, i god tid før møter.
Dårlig kommunikasjon innad i gruppen	Lav	Fører til uklare mål, dårlige prioriteringer og rutiner.	Ha fastsatte kommunikasjonskanaler og benytte disse jevnlig
Utstyr som ikke fungerer	Lav	Dersom utstyret over enn lengre periode er ødelagt/ubrukelig, vil	Raskt ta kontakt med aktuelle ansvarlig for vedlikehold av utstyr.

Vedlegg

Forprosjektplan

Prosjektnr

		det ikke være mulig å gjøre nødvendige målinger	Samle inn data fortløpende slik at man uavhengig av utstyr har noe data tilgjengelig.

6. Vedlegg

6.1 Tidsplan

Gantdiagram ligger vedlagt

6.2 Adresseliste

Firma: 3D Motion Technologies AS

Tlf: 915 49 447

Epost: post@3dmotech.com

Adresse: Sem Sælands vei 9, 7051 Trondheim

6.3 Avtaledokumenter

6.3.1 Arbeidskontrakt for bachelor-gruppen

Arbeidskontrakt ligger vedlagt

6.3.2 3-partsavtale

3-partsavtale ligger vedlagt

Vedlegg

Samtykkeskjema for innsamling av data

Bacheloroppgave
Oppmerksomhetsdeteksjon/-evaluering i VR med bruk av eye-tracking
3D Motion Technologies

Navn / Name: _____

Dataen som samles inn er anonym, og eneste potensielt identifiserbare data er avstand mellom øynene og bevegelsesmønster.
Jeg samtykker til at dataen samlet inn om min øyebevegelse brukes til forskning for 3D-Motion-Technologies, og eventuelt videre arbeid på dette prosjektet.

The data collected is anonymous, the only potential identifiable features are the distance between the eyes and the pattern of eye movement.
I consent to the use of the data collected from my eye movement for research purposes for 3D Motion Technologies, and potential further development of this project.

Ja / Yes

Nei / No

Andre / Other

Signatur tester Dato / Date

Signatur ansvarlig

Vedlegg

Systemdokumentasjon - prosjektets wiki

LogDataReader.md

"LogDataReader beskrivelsen er originalt skrevet av Marius Nygård og er hentet fra deres Gitlabside"

LogDataReader er en dataleser som implementerer grensesnittet `IClassifier`, og leser av tidligere blikkdata fra ei fil på datamaskina. LogDataReader er beregnet for bruk direkte i Unity Editor. Det er mulig å bytte ut fra hvilken loggfil man ønsker å lese fra direkte i Unity ved å klikke på Select File, som åpner opp en fillvalgmeny. For å opprette e `LogDataReader` bruker man konstruktøren

```
public LogDataReader (string datapath);
```

som tar en streng som argument. Argumentet er stien til ei loggfil. `datapath` kan være `null`.

For å hente data bruker man `GetData()`

```
public UnClassifiedData GetData ();
```

hvor `null` returneres hvis man ikke har ei fil å lese data fra. Hvert kall til `GetData()` leser av ei linje i loggfila.

Man kan hoppe til ei bestemt linje i loggfila ved å kalle `GoToLine (int value)`.

```
public void GoToLine (int line);
```

For å bytte hvilken loggfil man ønsker å lese fra, uten å gjøre det i Unity Editor, gjøres med metoden `OpenFile ()` og ved på forhånd sette den offentlig tilgjengelige variabelen `public string path`.

```
public void OpenFile ();
```

`OpenFile()` kaster `System.IO.IOException` hvis det oppstår en feil under henting av fil.

Datakollonner og -typer

Loggfilene som leses må være `CSV`-filer. De må ha følgende kolonner separert med `;` og datafeltene må fremkomme i rekkefølgen under:

#	Kolonnenanavn	Datatype	Beskrivelse
1	L_gaze_origin_mm	`Vector3`	En 3D vektor som beskriver posisjonen til venstre øye. På formen “(x,y,z)” hvor x,y og z er flyttal av typen `float`.
2	L_gaze_direction_normalized	`Vector3`	En 3D vektor som beskriver retningen venstre øye peker. På formen “(x,y,z)” hvor x,y og z er flyttal av typen `float`.
3	L_pupil_diameter_mm	`float`	Beskriver størrelse på venstre pupil i millimeter, på formen “x” hvor x er flyttal av typen `float`.
4	L_eye_openness	`float`	Beskriver hvor åpent venstre øye er, på formen “x” hvor x er et flyttal av typen `float` mellom 0.0f og 1.0f.
5	L_pupil_position_in_sensor_area	`Vector2`	En 2D vektor som beskriver posisjon til venstre pupil i sensorområdet. På formen “(x,y)” hvor x og y er flyttal av typen `float`. mellom 0.0 og 1.0.
6	R_gaze_origin_mm	`Vector3`	En 3D vektor som beskriver posisjonen til høyre øye. På formen “(x,y,z)” hvor x,y og z er flyttal av typen `float`.
7	R_gaze_direction_normalized	`Vector3`	En 3D vektor som beskriver retningen høyre peker. På formen “(x,y,z)” hvor x,y og z er flyttal av typen `float`.

Vedlegg

Systemdokumentasjon - prosjektets wiki

#	Kolonnenanavn	Datatype	Beskrivelse
8	R_pupil_diameter_mm	`float`	Beskriver størrelse på høyre pupil i millimeter, på formen “x” hvor x er et flyttall.
9	R_eye_openness	`float`	Beskriver hvor åpent høyre øye er, på formen “x” hvor x er et flyttall mellom 0.0 og 1.0.
10	R_pupil_position_in_sensor_area	`Vector2`	En 2D vektor som beskriver posisjon til høyre pupil i sensor området. På formen “(x,y)” hvor x og y er flyttal av typen `float` mellom 0.0 og 1.0.
11	C_gaze_origin_mm	`Vector3`	En 3D vektor som beskriver kombinert posisjon øye. På formen “(x,y,z)” hvor x,y og z er flyttal av typen `float`.
12	C_gaze_direction_normalized	`Vector3`	En 3D vektor som beskriver hvilke retning kombinert venstre og høyre øye peker. På formen “(x,y,z)” hvor x,y og z er flyttal av typen `float`.
13	Cam_Position	`Vector3`	En 3D vektor som beskriver posisjon av kamera i den virtuelle verden. På formen “(x,y,z)” hvor x,y og z er flyttal av typen `float`.
14	Cam_rotation	`Quaternion`	En quaternion som beskriver rotasjon av kamera i den virtuelle verden. På formen “(w,x,y,z)” hvor w,x,y og z er flyttal av typen `float`.
15	Timestamp	`int`	Et heltall som beskriver tiden data ble avlest, på formen “x”.
16	Frame_sequence	`int`	Et heltall som beskriver sekvens tallet til avlest data, på formen “x”.

Documents.md

On this page you will get a list of relevant documents used in this project. Expand the sections to view documents.

IDATT2501 - Road Sign Detection
Main report.pdf

IDATT2502 - Language Identification
Main report.pdf
Process report.pdf

Previous group's work
Main report.pdf
Litterature study.pdf

Friedman test
Test results.csv

Compiance form
Compliance form.pdf

Meeting 1 with Alexander Holt
Meeting report.docx
Meeting notice.docx

Meeting 2 with Alexander Holt
Meeting report.docx
Meeting notice.docx

Meeting 3 with Alexander Holt

Vedlegg

Systemdokumentasjon - prosjektets wiki

Meeting report.docx
Meeting notice.docx

Meeting 4 with Alexander Holt
Meeting report.docx
Meeting notice.docx

Meeting 5 with Alexander Holt
Meeting report.docx
Meeting notice.docx

Meeting with Ali Alsam
Meeting report.docx

Meeting with previous group
Meeting report.docx

Friedman-Test.md

Friedman test on different scenes

Scene	Sum	Average
Smooth-Movement-Move	12	1.7
Smooth-Movement-Still	22	3.1
Taxi-Tour-Move	40	5.7
Taxi-Tour-Still	21	3.0
Eye data 1	25	3.6
Eye data 3	44	6.3
Look and Remember	32	4.6

Future-Work.md

The work and the results achived with this project so far provide an optimistic starting point for further research on the eye tracking and related technologies. Firslty, the system can recognize the different types of scenes as well as who is using the HMD. This might seem like a relatively trivial issue. Nevertheless, it is important to look at what the results may imply and assess whether they are scalable to other problems. To be able to distinguish between who wears the HMD may be transferable to other research issues within the classification problem. One of these may be to classify diseases in the early stages, but it is to collect sensitive eye data from a large number of patients which might be a demanding and elaborate process. It is however deemed strictly necessary to explore the usability on a more general basis, before proceeding on such a project.

On the same topic of classifying diseases it could be interesting to look into datasets with regard to degenerative conditions of the central nervous system, as they are known to effect how a saccade performed. Using the saccade forecasting model maybe we would be able to say something on the behavior off the saccade, and what that can tell us about the test subject.

A concrete measure the group proposes is to make a larger data collection with more than three test subjects distributed among the new scenes created for this project. The results that many-ball-disappear could distinguish between test persons with very high precision, but with a limited data base this was not possible for the scenes one-ball-still and one-ball-move. Many-ball-disappear has variation linked to the user's reaction to the balls that disappear. In the one-ball-still and one-ball-move scenes, on the other hand, it is almost exclusively the users' still fixation and regular pursuits that will separate the eye data. If this is possible, it will be a big step to determine whether the accuracy of the eye tracking is sufficient to distinguish small details in the user's gaze.

Setup.md

Vedlegg

Systemdokumentasjon - prosjektets wiki

Clone project

Open terminal and write the following:

```
git clone git@gitlab.stud.idi.ntnu.no:bachelor102/unity-vr-bachelor.git
```

Dependencies

For Unity

For using unity project with eyetracking you need the following:

1. .NET Core
2. SteamVR
3. SRanipal
4. Windows 10 to run VR with SRanipal
5. NVIDIA 430.39 graphics driver (or newer) to run VR with eyetracking SRanipal
6. NVIDIA VRS 417.19 for SRanipal

Data Analysis

For data analysis you need to have python3 or anaconda installed.

Depending on what scripts you are going to run you may need the following python packages:

```
- pytorch
- torchvision
- pandas
- numpy
- tqdm
- matplotlib
- scipy
- scikit-learn
- tensorboard
- seaborn
- setuptools==59.5.0
- hmmlearn
- pyautogui
- opencv
- darts
- pillow
```

For setting up environent and with required dependencies simply run one of the following:

Pip

```
cd DataAnalysis2
virtualenv venv
source venv/bin/activate

pip3 install -r requirements.txt
```

Conda

```
cd DataAnalysis2
conda env create -f environment.yml
conda activate myenv
```

Vedlegg

Systemdokumentasjon - prosjektets wiki

Folder structure

Assets:

This folder contains main resources and scripts used in unity.

DataAnalysis:

This folder contains data analysis done by previous group.

DataAnalysis2:

This folder contains data analysis done by us. Contains some of the same scripts as from previous work, but unused scripts and files were removed and optimized.

LogDir:

This is the folder where raw data from running the scenes appear by default.

home.md

Welcome to Bachelor 102's Wiki page.

Here you will find our documentation of our project.

Navigation

Vision	Requirements	Documents	System
			
Future work	Setup	Documents	System
This page describes future work for this project.	An overview of the requirements nessesary to run the code and setup instructions for project.	Relevant documents for the final report of the main report.	Here you can find av overview of our system.

Question or need help?

Please contact us:

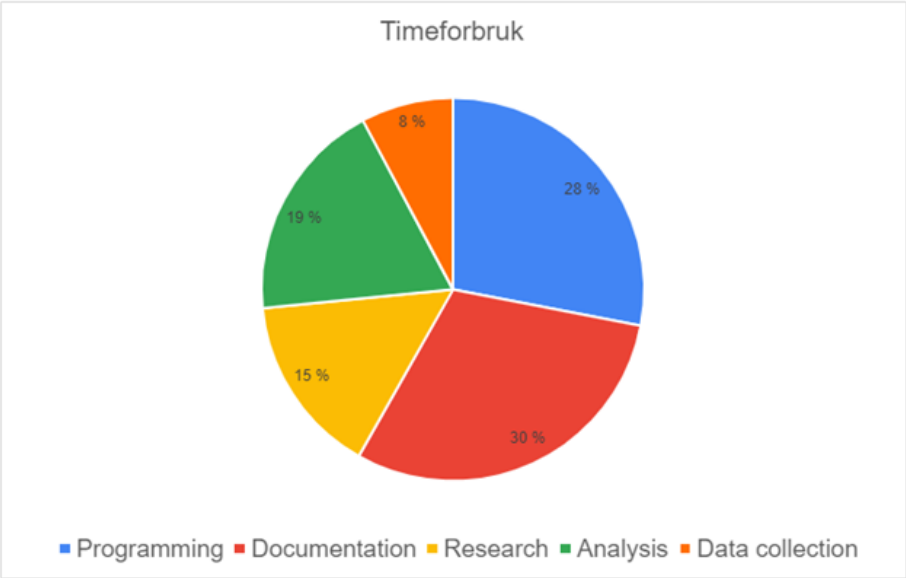
Lars Brodin Østby

Simon Jensen

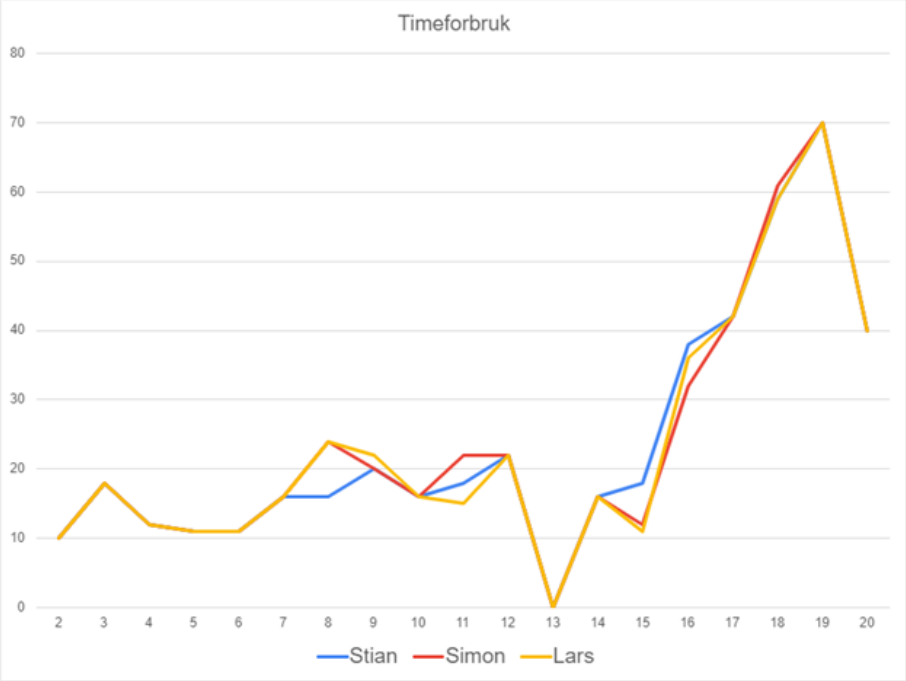
Stian Fjæran Mogen

Appendix

Tidsforbruk



Figur 68: Timefordeling per arbeidskategori



Figur 69: Timefordeling per uke