Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Birk Øvstetun Narvhus

Unsupervised klassifisering av øyebevegelser

Bacheloroppgave i Bacheloroppgave i Bachelor Ingeniørfag, data
Veileder: Alexander Holt

Mai 2024

NTNU
Kunnskap for en bedre verden

Birk Øvstetun Narvhus

Unsupervised klassifisering av øyebevegelser

Bacheloroppgave i Bacheloroppgave i Bachelor Ingeniørfag, data
Veileder: Alexander Holt
Mai 2024

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Sammendrag

Øyebevegelser kan kategoriseres i flere kategorier, for eksempel sakkader og fikseringer. Klassifisering av øyebevegelser er interessant i felt som psykologi og medisin. En lege kan for eksempel bruke øyebevegelse data for å se om en pasient har former for psykiske lidelser. Slik teknologi kan brukes til å advare brukere om lidelser, bare å benytte dagligdags teknologi. Digitale klokker som måler blodtrykk og hjerterytmer er eksempler på lignende teknologi.

Problemet med dagens løsninger er at de er dyre og u-presise. Bruk av ir-kamera og tradisjonelle algoritme baserte modeller er vanlig. Algoritmene må også kalibreres med terskelverdier eller lignende. Hvis teknologien skal brukes i konsumer produkter, må nye løsninger utvikles. Dette er to av flere utgangspunkt bak problemstillingen, som skal besvares gjennom rapporten.

Resultatet av tradisjonelle algoritmer, vil gi forskjellige resultater avhengig av spesialisten som har kalibrert algoritme. Klassifisering av øyebevegelser er ikke gjort ved en universal utformet standard. Hva som regnes som en øyebevegelse vil variere fra person til person, og flere av bevegelsene kan ha overlapp i hva som kjennetegner dem. I prosjektet vil det bli utarbeidet en bedre modell, som ikke er påvirket av menneskelig interferens. Metodene self-supervised og un-supervised maskinlæring blir brukt for å håndtere dette problemet.

Gjennom prosjektet utforsket jeg en rekke forskjellige modeller. Den endelige modellen var en transfer-learning basert modell, som brukte en self-supervised auto-enkoder og un-supervised en dim-reduksjon + k-means klynge modell. Klassifiseringene modellen gjorde er basert på sammenhenger som ble uthentet uten menneskelig innflytelse.

Resultatene svarte ikke på problemstillingen. Ende-til-ende modellen greide ikke klynge øyebevegelsene i meningsfulle klynger. Modellen jeg brukte var utilstrekkelig for oppgavens vanskelighetsgrad. Ved videre arbeid anbefales bruk av andre modeller, for eksempel en Disentangling Variational Autoencoder, eller revurdere bruk av en unsupervised ende-til-ende modell.

Forord

Jeg valgte denne oppgaven, ettersom den omfanget interessante temaer innenfor maskinlæring og dyp læring. Jeg hadde en grei forståelse for grunnleggende metode i maskin læring før jeg gikk inn i prosjektet. Oppgaven er krevende, men potensialet for læring var godt. Jeg valgte dermed oppgaven, da jeg kunne lære nye, og mer rettede metoder innenfor maskinlæring. I tillegg handlet oppgaven om temaet øyebevegelser. Temaet var mindre utforsket, spesielt på maskinlærings fronten, og derfor virket interessant å utforske noe som aldri har blitt testet.

I prosjektet implementerer jeg flere ende-til-ende modeller for klassifisering og klynging av øyebevegelser. Målet er å kunne lage en modell som gir gode resultater uten ekstern menneskelig interferens ved klassifisering. En god modell kan fremme forskning innenfor øyebevegelser. I tillegg kan en god modell være med på å utvikle ny VR teknologi, som vil gjøre det mer realistisk.

Jeg vil takke veileder og oppgavegiver Alexander Holt, for å ha hjulpet med veiledning og ekstra datakraft gjennom Vizlab. Jeg vill også takke Alexander Holt for å gi åpne tøyler rundt problemstilling, så jeg kunne utforske i den retningen jeg syntes var mest interessant.

20.05.2024 - Trondheim

Birk Øvstetun Narvhus

Innhold

Figurer

Tabeller

1. Introduksjon

Deteksjon av øyebevegelser har lenge vært et tema. I 1879 gjorde Louis É. Javal et eksperiment for å observere øyebevegelser og klassifisere dem (Wade, 2010). I undersøkelsen brukte de speil for å se på øynene. Senere i 1908 lagde Edmund B. Huey den første maskinen for å observere øyebevegelser (Huey, 1908). I dag brukes ofte små kameraer som kan spore øynenes bevegelser. Dette blir for eksempel benyttet i moderne vr headset.

Klassifikasjoner av øyebevegelser kan brukes i medisinske applikasjoner. Psykiske lidelser påvirker øyets bevegelser(Bittencourt J., 2013), ved å analysere øyebevegelsene kan leger få et bedre innblikk på slike lidelser. Ved gode fremtidige løsninger kan dagligdags teknologi brukes for å advare brukere om helsetilstander. Vi ser allerede slik teknologi i for eksempel digitale puls klokker.

Dagens metoder for blikk deteksjon bruker kameraer (ofte infrarøde) og forskjellige algoritmer for å kunne hente ut data som virker hensiktsmessig. Vanligvis brukes spesialist utstyr som er både dyrt, og vanskelige å sette opp (Mestre C, 2018). Slike metoder gjøre det mindre hensiktsmessig å bruke i for eksempel vr. Nyere metoder bruker maskin læring for å hente ut denne dataen (Zhao L, 2018). Problemet med begge metodene er at de kun henter ut data som vi tror er viktig. For eksempel, pupille vinkelhastighet og pupillstørrelse.

I problemstillingen skal jeg prøve å håndtere dette problemet, ved å gå fra øyevideo og øyebevegelse klassifikasjon i en modell. Dette vil gi modellen flere faktorer å ta hensyn til ved klassifikasjon. I tillegg vil modellen være unsupervised. Ved å ikke bruke annotert data for læring, vil modellen kunne finne nye sammenhenger. De nye sammenhengene kan gi et nytt syn på hvordan vi skiller øyebevegelser.

Oppgaven går ikke ut på å lage en modell som kan konkurrere med eksisterende algoritmer. Prosjektet vil være en test om automatisk feature uthenting ved en ende-til-ende modell kan være et alternativ til tradisjonelle algoritmer. Et av kravene for gjennomføring av problemstillingen vil være at resultatene ved testing av modellen skal overlappe med resultatene fra tradisjonelle algoritmer. I tillegg skal det ikke være menneskelig interferens i noen av modellens ledd. Til slutt, modellen skal bruke videosekvenser av øyebevegelser.

Problemstillingen over var utarbeidet og endret gjennom prosessen, ettersom tester og ny kunnskap avslørte hva som var overkommelig. Der det er gjort endringer i problemstilling, er dette påpekt ved gjennomgang av metode. Den originale problemstillingen er vedlagt.

Gjennom prosjektet ble det gjennomført flere tester og dermed nye iterasjoner av modellene. Metode kapittelet går dypere inn på valg og tester som ble gjort underveis. Prosjektet ble gjennomført med hensyn på slik utforsking, istedenfor et sluttresultat. Resultater underveis og antagelser med teoretisk grunnlag, blir derfor vektlagt i rapporten.

Self-supervised og un-supervised maskinlæring er essensielt i implementasjonen. Forskjellige teknikker innenfor feltene blir utypet i teori kapittelet, men det er forventet en grunnleggende forståelse for maskinlæring.

2. Teori

I teorien skal jeg gå gjennom relevant teori som må kunnes for å kunne gjennomføre oppgaven. Teorien er for de meste rettet mot maskinlæring, men jeg skal gå gjennom grunnleggende teori innenfor øyebevegelser, og øyebevegelses data. Dette vil være relevant for å forstå hvorfor jeg har valgt de metodene jeg har.

2.1 Øyebevegelser

Øyebevegelser er sentrale for hvordan øynene beveger seg mellom interessepunkter. For å se hvorfor slike bevegelser er viktig må vi se på øyets anatomi. Delen av øyet som henter inn informasjon for sentrum av synsfeltet, heter fovea. Foveaen har bare 0.35 millimeter diameter, men er fortsatt punktet med høyest visuell skarphet (Rehman I, 2023). Synsfeltet man får ved bruk av den beste visuelle skarpheten i foveaen er ca. 5 grader (Strasburger, 2020). Dermed er det viktig for øye å kunne bevege seg, for å kunne hente inn interessepunkter.

Bevegelsene til øyet er delt opp i fire hovedkategorier. Kategoriene er sakkader, jevn forfølgelsebevegelser, vergensbevegelser og vestibulo-okulære bevegelser (Purves D, 2001b). Bevegelsene kan flytte sentralblikket til nye interessepunkter, eller motvirke andre bevegelser for å holde blikket på eksisterende interessepunkter. Det er også flere underkategorier rundt de nevnte hovedkategoriene, men de er ikke relevante for oppgaven. Jeg går ikke dypere inn på u-frivillige øyebevegelser som forekommer under medisinske årsaker, ettersom dette heller ikke er relevant.

I tillegg til bevegelsene nevnt over, er fikseringer sentralt i denne oppgaven. Om fikseringer er en øyebevegelse sammenlignet med de andre kan diskuteres. Fikseringer går ut på hvordan øyet holdes stille for å fokusere på et spesifikt interessepunkt. Dette er ikke en bevegelse direkte, men det er fortsatt viktig når vi skal klassifisere bevegelsene. Ved naturlig bruk av øynene, er det fiksering som forekommer mest. Spesielt Hvis man måler øyebevegelser over tid (Alexander mfl., 2018).

2.1.1 Sakkader

Sakkader er hurtige bevegelser av øynene, som kjapt endrer blikket sitt interessepunkt. Slike bevegelser kan variere fra større bevegelser når man

for eksempel ser etter noe, til små bevegelser når man leser en bok (Purves D, 2001a). Hoved aspekter for en slik bevegelse er hastigheten til pupillen. En sakkade bevegelse har en topp hastighet normalt fordelt fra 281 til 541 med et gjennomsnitt og standardavvik på 393 +- 50 grader/-sek (EL, 1985). Hastigheten ble målt over en 20 grader rotasjon av øyet. I undersøkelsen i EL, 1985, observerer de at sakkadiske bevegelser som har en hastighet under 300 grader pr sekund er unormalt lavt. Dette er et viktig punk når vi skal bruke tradisjonelle algoritmer ved klassifisering av slike bevegelser.

I tillegg varer sakkader vanligst mellom 35 – 50 ms, og har et gjennomsnitt på 40 ms (Collins mfl., 2008). For enkelte tradisjonelle algoritmer er tid og varighet viktig, for eksempel i I-DT algoritmen. Ved maskinlæring er dette også sentralt, ettersom vi må vurdere hvor store data som er hensiktsmessig for en god klassifikasjon.

2.1.2 Post-sakkadisk drift og Dynamisk overskridelse

Post-sakkadisk drift og dynamisk overskrid er begge bevegelser som skjer etter en sakkadisk bevegelse. Dynamisk overskridelse er når øyet bremser en sakkade, men overkorrigerer bremsingen, og dermed må øye korrigere igjen med en liten sakadisk bevegelse for å rette feilen (Kapoula mfl., 1987). Ved post-sakkadisk drift beveger øyet seg med lavere hastighet. Drift bevegelsen har ikke samme trekk som en vanlig sakkade ved at hastigheten er lav. Slike drift bevegelser er vanskelig å differensiere fra andre øyebevegelser. Begge er korte i tid og påvirkning. Dermed velger jeg å ikke ta hensyn til dem ved evaluering av modellen.

2.1.3 Jevn forfølgelse bevegelser

Jevn forfølgelses bevegelsers er når øyet beveger seg sakte for å forfølge en interesse punkt. Det er ikke uvanlig at man i stedet for å bruke jevn forfølgelse bevegelse, bruker man heller en sakkade. Spesielt når interessepunktet ikke beveges relativt til sentral synet (Purves D, 2001a). Bevegelsene kan variere i hastighet, Wong, 2008 viser til jevn forfølgelse bevegelser fungerer bra opp til 70 grader pr sek og trente individer opp til 130 grader pr sekund. Hastigheten på jevne forfølgelse bevegelser kan variere fra 0 til 70-130 grader pr sekund. Ved overskriding vil man bruke små sakkader.

Varigheten til jevn forfølgelse bevegelser varierer. Først vil øynene akselerere i ca. 100 ms deretter vil øynene følge interesse punktene med en konstant fart (Krauzlis & Lisberger, 1994). Å klassifisere øyebevegelsene

under akselerasjon kan være vanskelig, ettersom vi vil klassifisere dem sammen med bevegelser som allerede har nådd forfølgelses hastigheter.

2.1.4 Vergensbevegelser

Vergensbevegelser er øyebevegelser hvor øynene beveger seg i forskjellige retninger. Dette kan for eksempel være når man ser på noe veldig nærme, der øynene vil konvergere. På andre siden hvis øynene går fra hverandre vil øynene divergere, en slik bevegelse er også en vergensbevegelse (Purves D, 2001a). Hastigheten til vergens bevegelser avhenger av forskjellen i hastighetene mellom hver pupille. Dette kan for eksempel være hvis et øye har en sakkade som er kjappere enn det andre (Koken & Erkelens, 1994).

2.1.5 Vestibulo-okulære bevegelser

Vestibulo-okulære bevegelser er en refleksbevegelse, altså kan ikke gjennomføres med vilje, og benyttes for å kompensere for hodebevegelser. Slike bevegelser er grunnen til at man kan lese når man for eksempel går langs gaten. Bevegelsene stabiliserer blikket selv om hode har en vinkelhastighet på over 300 grader pr sekund. Slike hastigheter er sammenlignbare med sakkadiske bevegelser. I tillegg, siden slike bevegelser er reflex, kan man utføre dem opp til 20 ganger pr sekund (Broussard mfl., 2010). Den korte reflex forsinkelsene er en av måtene vanlige algoritmer ser forskjellen mellom slike bevegelser og for eksempel sakkader.

2.1.6 Fiksering

Fiksering er når øynene slutter å søke etter interessepunkter og låser seg til et punkt. Dette er dermed ikke en øyebevegelse i seg selv, men under fiksering bruker øyet fikserende øyebevegelser. Disse er skjelving (den minste), drifter og mikrosakkader (den største) (Martinez-Conde S, 2004). Drift fikseringer har en hastighet på under 2 grader / sekund, og forekommer etter microsakkader (på lik linje som post-sakkadisk drift) (Alexander mfl., 2018). Alle fikserings bevegelser er relativt små bevegelser, og har en kort varighet.

Ved klassifisering er det viktig å ta med fikseringer, ettersom 80% av blikk vil være fikseringer (Alexander mfl., 2018). Dermed vil dette også få stort utslag ved klassifikasjon. I tillegg er fikseringer gjennomsnittlig den lengste øyebevegelsen/blikk. Lengdene på fikserings blikk er ikke normalt fordelt, med en median på 200-250 ms og en gjennomsnittlig lengde på 300-350 ms (Negi & Mitra, 2020). Fikseringer varierer stort i lengden. En-

kelte fikseringer kan være 50 ms, men på andre siden har du fikseringer som kan vare mer en 2000 ms.

2.2 Tradisjonelle algoritmer

Det er flere forskjellige eksisterende algoritmer. De varierer på hvordan de oppnår en klassifikasjon og hvliket data de trenger for å gjennomføre klassifikasjon. I-VT (Velocity threshold identification) og I-DT (dispersion threshold) er eksempler på slike algoritmer (Salvucci Dario D., 2000). Begge algoritmene klassifiserer fikseringer og sakkader, men de klassifiserer ikke jevn forfølgelse bevegelsen. På andre siden har man for eksempel ivvt algoritmen. Denne kan også klassifisere jevn forfølgelse bevegelsen (Birawo Birtukan, 2022). I tillegg til de nevnte algoritmene, er flere andre lignende algoritmer. Ettersom oppgaven ikke går ut på evaluering av slike modeller, går jeg dypere inn i de overnevnte algoritmene.

2.2.1 I-VT (Velocity threshold identification)

I-VT modellen bruker pupillens vinkelhastighet og en terskel verdi for å kunne bestemme om en bevegelse er sakkada eller fiksering. En slik modell kan gi gode resultater. Problemet er at denne terskelverdien kan være vanskelig å finne, avhengig av blant annet hvordan man henter/kalkulerer hastigheten (Birawo Birtukan, 2022). Salvucci Dario D., 2000 foreslår en terskelverdi på 20 grader pr sekund, men dette må kalkuleres med hensyn på opptaks raten på datasettet. I tillegg kan modellen kun klassifisere fiksering og sakkader, ettersom det er kun en terskelverdi. På andre siden kan man da bruke en IVVT algoritme.

2.2.2 I-DT (Duration threshold identification

I-DT modellen er en fordelings basert algoritme, og bruker dette for å klassifisere sakkader og fikseringer. Ettersom fikseringer ikke beveger mye i forhold til sakkader, vil man kunne se på om mange punkter havner i samme område over tid. Fikseringer varer ofte i over 100 ms. Dermed kan man klassifisere en bevegelse som fiksering hvis flere pupille posisjoner oppholder seg i samme område over et tidsrom. Tidsrommet og mengdens fordeling vil da være terskelverdien (Salvucci Dario D., 2000).

2.2.3 IVVT

Algoritmen ligner I-VT algoritmen, men det er en ekstra pupille vinkelhastighet terskelverdi. I motsetning til I-VT og I-DT kan algoritmen også

klassifisere jevn forfølgelse bevegelse. Først vil algoritmen bestemme om det er en sakkader eller fiksering ved å bruke I-VT algoritmen. Etter den har bestemt om det er sakkade eller fiksering vil den bruke en ny terskelverdi og klassifisere jevn bevegelse (Birawo Birtukan, 2022). Ettersom jevn bevegelse har en hastighet på ca. 0-70 grader pr sekund (Wong, 2008), kan man sette en terskelverdi på sakadene (fra I-VT algoritmen) å klassifisere jevn forfølgelse bevegelse. På den andre siden, algoritmen er ikke feilfri ettersom det er stor variasjon i terskelverdiene, men dette er et problem alle terskelverdi algoritmer vil ha.

2.2.4 Random-forest og andre ml-metoder

I likhet med de allerede nevnte algoritmene, vil eksisterende maskinlærings baserte metoder ikke være ende til ende. Dette betyr at eksisterende modeller ikke bruker fulle bildesekvenser for klassifisering. På andre siden, slike modeller har vist bedre resultater enn eksisterende algoritmebaserte metoder (Zemblys R., 2018). De bedre resultatene får slike modeller ved å redusere menneskelig input. I maskinlærings modeller trenger man for eksempel ikke sette terskelverdier. Modellene må fortsatt trenes med annotert data, som betyr at det ikke er unsupervised.

2.3 Databehandling

Ettersom de eksisterende modellene ikke er ende til ende, trenger vi en algoritme/modell for å hente ut nyttig «features» fra datasettet. Grunndataen for øyebevegelsene vill være videoer tatt av øynene. Algoritmene jeg har nevnt trenger en form for feature uthenting for å kunne brukes. I for eksempel I-VT modellen, brukes pupillens vinkelhastighet for klassifisering. For å kunne hente ut vinkelhastigheten på pupillen fra grunndataen, må man bruke en øye-sporer (Eye-tracker).

Øye sporings metoder kan variere i kostnad og treffsikkerhet. Moderne metoder bruker for eksempel hornhinnene refleksjon for å hente ut pupille posisjon (Tobii, 2024). En slik metode er dyr i praksis og lite hensiktsmessig for problemstillingen. En annen metode er bruk av maskinlæring for øye sporing. Google MediaPipe Iris er et eksempel på en slik modell (Andrey Vakunov, 2020). Denne modellen vil være billig å implementere, og vill gi middels gode resultater (i forhold til hornhinne refleksjons metoden). Til slutt, enkle kontrast baserte algoritmer kan også brukes. Slike algoritmer fungerer ved å se på kontrasten mellom hornhinnen og området rundt. Kontrast baserte algoritmer gir relativt gode resulter på infrarød data, ettersom kontrasten vill være høyere. En slik modell er lett å implementere, men er mindre nøyaktig enn andre populære metoder.

En fellesfaktor for alle 3 metodene nevnt ovenfor, er at de trenger menneskelig tilsyn ved implementering. I googles maskinlærings metode, bruker de annotert data for å trene opp modellene (supervised læring) (Artsiom Ablavatski, 2019). De to andre modellene får menneskelig tilsyn ved for eksempel terskelverdier og kalibrering. Problemstillingen påpeker at modellen skal være uten noen form for menneskelig interferens, dermed må vi se dypere på unsupervised læring.

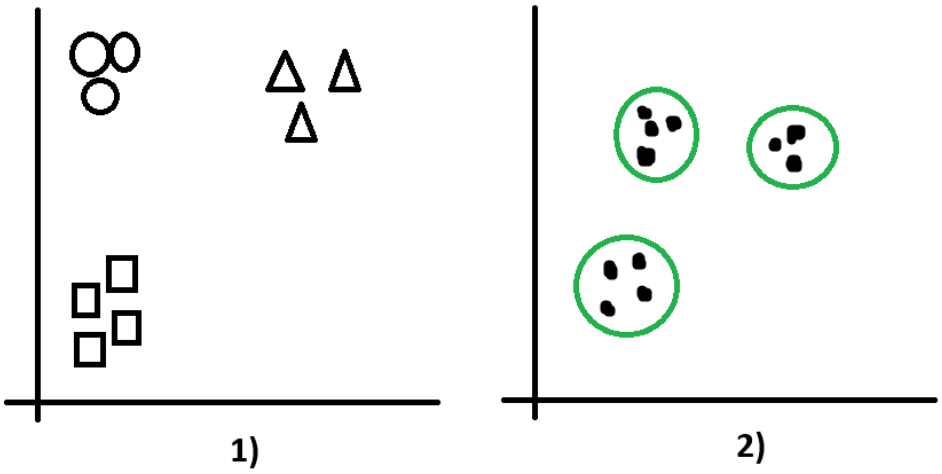
2.4 Unsupervised maskin læring

Unsupervised læring eller læring uten tilsyn er maskinlæring uten noen form for menneskelig tilsyn ved trening. Der supervised lærings modeller trenger annotert data, bruker unsupervised læring data uten slik annotering. Dermed vil modellen kunne finne mønster og trekk ved datasettet den mener skiller dataen fra hverandre (GoogleCloud, u.å-b). Ønskelig vil modellen finne trekk som vi først ikke tenkte var essensielle for klassifisering, og dermed gi oss et nytt innblikk. En slik modell passer problemstillingen, ettersom vi ønsker å finne en universal «ground trouth» uten noen from for menneskelig innsikt. Et slikt resultat ville ikke vært mulig ved vanlige maskinlæring/algoritme modeller. Modellen må være en såkalt clustering modell.

2.4.1 Clustering eller klassifisering

Ved bruk av eksisterende modeller klassifiserer vi data inn i bestemte kategorier. Ved øyebevegelser vil dette for eksempel være sakkader, fikseringer, jevn forfølgelse bevegelse osv. Dette gjør modellen ved å lære sammenhengen mellom en annotering og dataen den omfavner. En trent klassifiserings modell vil forhåpentligvis kunne klassifisere ny data. Klassifiserings metoden vil da etterligne den menneskelige klassifiseringen. Resultatet vil variere etter kvaliteten på dataen som brukes og annoteringen. Her ligger samme problemet som de vanlige terskelverdi baserte modellene, ved at resultatet kan variere avhengig av hvem som bestemmer data annoteringen.

På andre siden har man clustering. Clustering modeller ser på forskjellen mellom datapunktene og prøver å gruppere punktene ved hjelp av disse sammenhengene. Dermed vil ikke modellen etterligne annoteringen, men finne nye mønster eller trekk ved dataen (GoogleCloud, u.å-a). Med øyebevegelser som et eksempel, ville modellen kunne se at to sakkader ligner mer på hverandre enn en fiksering. Det er en rekke forskjellige clustering metoder, som har varierende bruksområde. Jeg skal gå dypere inn på metodene som var relevant for oppgaven.



Figur 2.1: 1) Eksempel på en klassifiserings distribusjon. Her har alle datapunktene en annotering. **2)** Eksempel på en klase distribusjon. Her er lignende punkter gruppert

2.4.2 Clustering metoder

Det er fire hovedkategorier innen clustering: eksklusiv clustering, overlappende clustering, hierarkisk clustering og probabilistisk clustering (GoogleCloud, u.å-b). Metodene har forskjellige bruksområder, og gir forskjellige resultater som passer til disse. Under eksklusiv clustering har man for eksempel k-means. Med hensyn på oppgavens størrelse har jeg bestemt å gå nærmere inn på akkurat denne metoden.

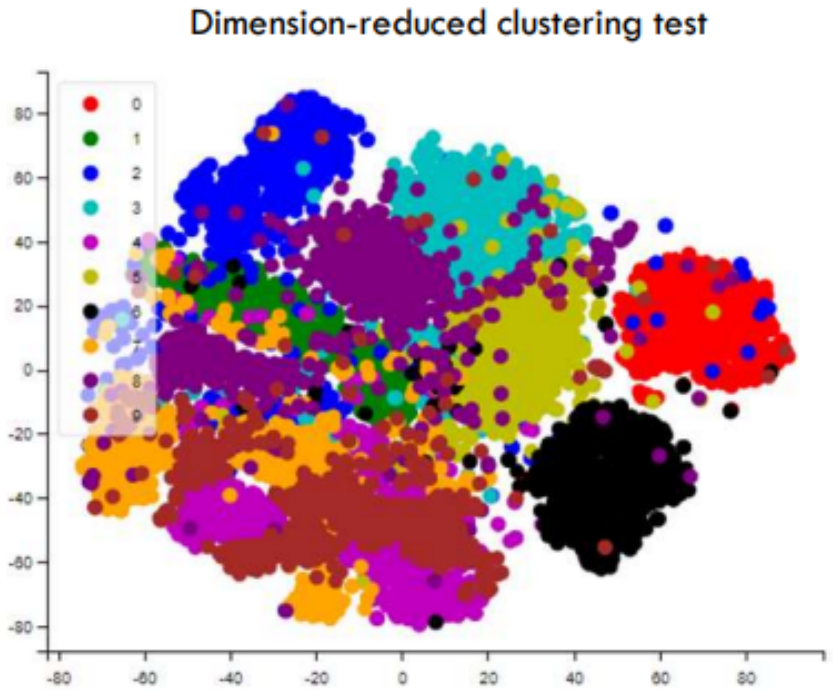
K-means

K-means algoritmen inngår under eksklusiv clustering eller hard clustering. Dette betyr at hvert datapunkt bare kan havne i en kategori. Under k-means er det også mange forskjellige varianter, for forskjellige bruk (Ahmed mfl., 2020). På grunn av oppgavens bredde går jeg inn på hvordan vanlig k-means fungerer. En naiv k-means algoritme går ut på å først sette k tilfeldige sentrums punkter. Deretter klassifisere datapunktene mot de senter punktene som er nærmest i Euklidisk distanse. Her kan punktene bare klassifiseres mot et senterpunkt. Deretter oppdateres senterpunktene ved å flytte dem til midtpunktet til mengden midtpunktet klassifiserer. Deretter vil denne prosessen oppdateres til senterpunktene ikke oppdateres lengre (Steinley, 2006). Bemerk at dette er en naiv forklaring av k means, ettersom det er bedre og raskere metoder. På den andre siden vil alle metodene fungere prinsipielt likt.

Naive metoder som enkel k-means er ikke gode nok i seg selv til å kunne klassifisere øye bevegelses data. Ingen un-supervised klynge algoritme vil gi gode resultater på høy dimensjons data.

2.4.3 Dimensjon reduksjon

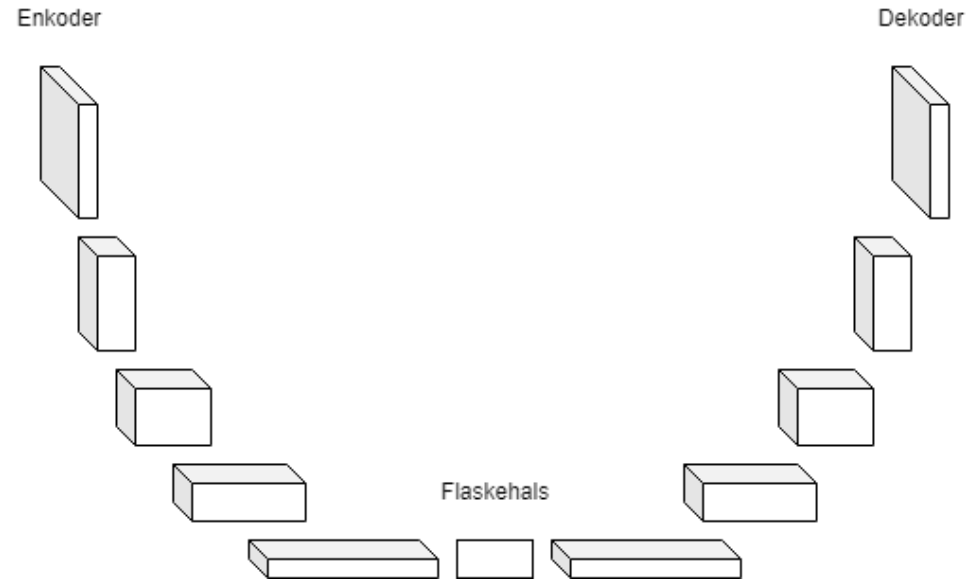
Dimensjon reduksjon er forskjellige teknikker som brukes for å forbedre transfer læring mellom modeller. Dette gjør det ved å minke spredningen av denne dataen ved å minke dimensjonaliteten til dataen (Salih Hasan Basna Mohammed, 2021). Data kompresjon er et eksempel på slik dimensjon reduksjon. I tillegg til å gi bedre resultater vil også slike teknikker gjøre det lettere i et hastighets perspektiv. For eksempel å trene en tidskrevende algoritme som k-means er lettere på 3-dimentional data enn om dataen har 200 dimensjoner. I figur 2.2 ser man et eksempel på dimensjon reduksjon uten å miste relevant context. I figuren er metoden t-sne bruk, i tilegg til denne metoden har man for eksempel PCA og auto-encoders.



Figur 2.2: Eksempel på dimensjons redusert data fra MNIST-datasettet (datasett fra Li., 2012)

Pca (Principal Component Analysis)

Pca er en teknikk for å redusere dimensjonaliteten til dataen, samtidig som variansen maksimeres. Dette gjør den ved å redusere punkter ned til flere «pricipal axis» eller hovedakser (Kurita, 2014). Denne metoden blir ofte brukt i mønstergjenkjenning, i for eksempel ansikts gjenkjenning (Kurita, 2014). I Mwanga mfl., 2023 viser de at ved å bruke slik dimensjon reduksjon økte de treffsikkerheten drastisk, spesielt i ikke-dyp lærings modeller. I tillegg viste de til at slik dimensjon reduksjon reduserte treningstiden og kjøretiden for modellen. På den andre siden, er ikke pca en feature uthentigs teknikk (de-la-Bandera I., 2020), dermed må vi også bruke en auto-encoder.



Figur 2.3: En visuell representasjon over hvordan en auto enkoder kan se ut. Enkoder, Dekoder og Flaskehals markert. Størrelsene på cubenes lengde og høyde er proporsjonale med bildestørrelsen, og bredden er antall kanaler pr bilde. Flaskehalsen er flat ettersom det skal symbolisere et fully-connected lag

Auto-enkoder

En auto enkoder er en modell som er bygd for å gjenskape opprinnelig inn data. Dette gjøres ved å bruke en enkoder for å først bruke feature uthenting og dimensjons reduksjon. Resultatet av denne enkoderen er resultatet vi er ute etter, ettersom det oppfyller både kravene om feature uthenting, som er å sette sammen gamle featuers til nye, og dimensjons reduksjon, som er å redusere dimensjonaliteten uten å miste kontekst (Zhai mfl., 2018). Jeg skal gå mer inn på enkoderens og dens oppbygning senere under generell maskinerings kapittelet, ettersom dette er den mest

essensielle delen for et godt resultat. I tillegg til enkoderen har man en dekoder og en valgfri flaskehals. Funksjonen til dekoderen er å reprodusere originale bilder ut ifra flaskehalsens resultat. En auto enkoder ligner i stor del på et u-net, som ofte blir brukt i bilde generering og bilde segmentering (Croitoru mfl., 2023; Ronneberger mfl., 2015). Den største forskjellen er at auto enkodere bruker ikke skip-connections mellom lagene og u-net bruker supervised læring.

Contrastive modeller

Til slutt, grunnen til at man bruker en enkoder og en dekoder, er fordi vanlige taps funksjoner må sammenligne noe med resultatet til modellen. I auto enkoderen oppnår man dette ved å sammenligne original dataen med predikert data. Problemet med en slik modell er at man må doble modelldybden. Dette betyr at dataen må først gå ned enkoderen, og deretter opp dekoderen, før det blir gjort en prediksjon. Dypere modeller er vanskeligere å trene enn mindre dype modeller, fordi man møter forsvinnende/eksploderende gradientproblemet (He mfl., 2015). Problemet oppstår når for mange lag påvirker gradienten, spesielt enkelte aktiverings funksjoner. I tillegg kan man ikke bruke en stream-buffer som er en metode der man reduserer maksimalt minnebruk. Jeg skal komme nærmere på hvorfor en slik buffer er gunstig senere i rapporten.

På den ander siden, hvis man bruker en kontrastiv modell, trenger man ikke bruke dekoderen i auto-enkoderen. Først vil man sende dataen gjennom to separate og tilfeldige data behandlinger. Deretter vil man sitte med to forskjellige, men korrelerende data eksempler. Dataen vil bli sent gjennom modellen i en separat passering for hvert data eksempel. Den reduserte dataen fra modellen vil da sammenlignes med hverandre i en spesiell taps funksjon. Denne taps funksjonen har som jobb å maksimere korrelasjonen mellom begge data eksemplene (Ting Chen, 2020). I Ting Chen, 2020 beskriver de at en slik semi-supervised modell er kritisk avhengig av tilfeldig datatransformasjon, større batch størrelser og lengre trenings varighet. Dette er i forhold til lignende supervised lærings baserte modeller.

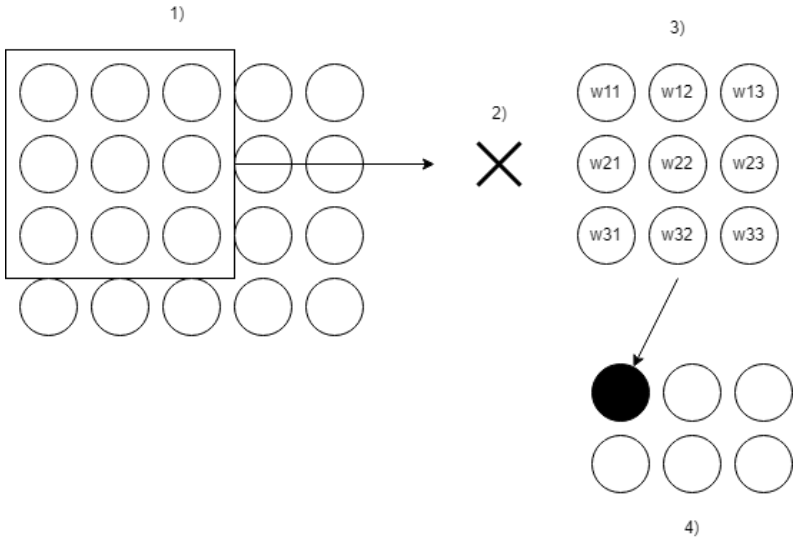
Selv om en velger en slik kontrastiv modell eller en auto-enkoder, vil en convolution basert feature uthentings enkoder være essensielt.

2.5 Maskin Læring

Maskinlæring kan gjøres på en rekke forskjellige måter. I dette kapitelet skal jeg gå gjennom viktige deler som var essensielt ved oppbygningen av

modellen jeg brukte. For å holde meg innenfor oppgavens grenser, vil jeg ikke gå dypt inn på for eksempel backpropagation og aktivasjons funksjoner. Metodene jeg skriver om er derimot større overordnede konsepter. Disse konseptene vil være viktige for en overordnet forståelse for, hvorfor jeg gjorde valgene jeg gjorde, under metode gjennomgang.

2.5.1 Convolution baserte nettverk



Figur 2.4: 1) input data punkter **2)** Element-vis multiplikasjon med filter, og akkumulering **3)** Filter med vekter **4)** output data punkter

Convolution baserte nettverk ofte kaldt «CNN» brukes vanligvis til for eksempel bilde klassifikasjon (Neha Sharma, 2018). En slik cnn er god på uthenting av abstrakte sammenhenger i daten (Sakshi Indolia, 2018). I følge Sakshi Indolia, 2018 er dette fordi, cnn-er bruker delte vekter, dermed vil mengden parametere reduseres stort i forhold til for eksempel linjere nettverk. Ved å redusere parameter mengden vil risikoen for overfitting reduseres. I tillegg vil de færre parameterne gjøre det mulig å lage dypere nettverk. Dypere nettverk har vist bedre evnen ved bilde klassifisering enn mindre dype modeller. For eksempel i Sarwinda mfl., 2021 gir resnet-18 modellen generelt dårligere resultater enn resnet-50 modellen.

CNN nettverk er ikke alle bygd opp på samme måte, der det er flere forskjellige metoder innen convolution. Metodene har forskjellige bruksområder, og har også forskjellige styre og svakheter som korrelerer med bruksområdene. I den modellen jeg har foreslått brukes en kombinasjon av Vanlig convolution, dilated convolution og causal convolution. I tillegg vil convolution dimensjonen variere med data dimensjonen. Dette kan være

for eksempel 2d eller 3d convolution.

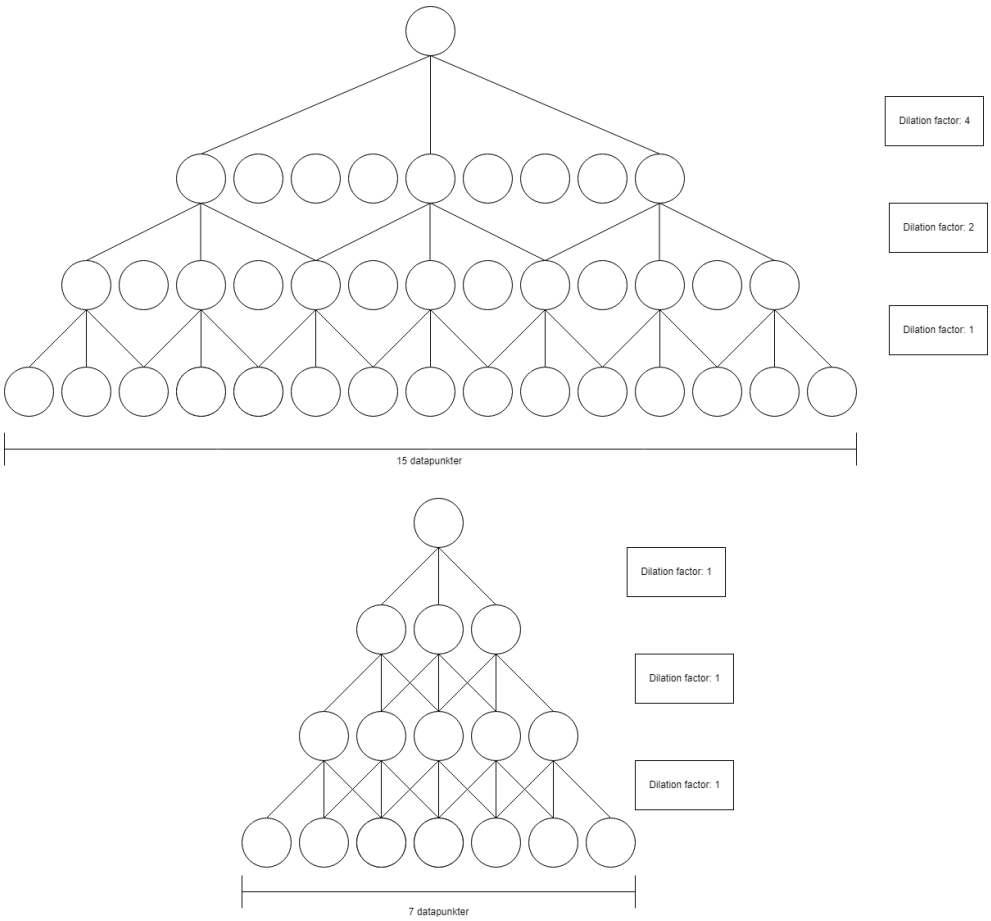
2.5.2 Vanlig convolution

Convolution er en maskinlærings metode, der man bruker en fast kernel/-filter og sklir det over dataen og bruker en convolution operasjon (Sakshi Indolia, 2018). Figur 2.4 er en visuell representasjon av hvordan dette kan se ut. I figuren er det en 3x3 filter størrelse og ingen padding. Padding kan man legge til før man kjører convolution, for å endre størrelsen på resulterende data. I tillegg kan man variere striden, eller hoppene, som filteret gjør over dataen. Dette ser man i 2.5, der øking av stride vil proporsjonalt minke data størrelsen etter convolution.

$$\text{size}_{\text{out}} = \frac{\text{size}_{\text{in}} + 2 * \text{padding} - \text{kernel}}{\text{stride}} + 1$$

Figur 2.5: Data størrelsen ut av convolution avhenger av størrelsen inn, padding, kernel/filter størrelse og stride

Fordeler med vanlig convolution er at de er gode på å hente ut lokale sammenhenger og mønsteret (Yamashita R., 2018). For å sette dette i perspektiv; et linjert lag der alle nodene er koblet sammen. I et slikt lag vil modellen kunne hente inn globale sammenhenger etter første lag. På andre siden må convolutionlag ha flere lag satt oppå hverandre, sammen med downsampling teknikker for å finne samme globale de sammenhengene. Et slikt lagbasert nettverk vil tvinge modellen til å finne abstrakte sammenhenger, og dermed unngå overfitting. På den andre siden, en slik modell vil gi dårlige resultater der sammenhengene er globale over dataen. For eksempel ved oversettelse. I Tang mfl., 2018 ser man at cnn baserte modeller, gir dårligere resultater enn RNN (rekursive modeller) og Self-attention modeller (transformer modeller). En måte å motvirke dette problemet, er ved bruk av dilated convolution.



Figur 2.6: Syns felt for convolution med og uten skalerende dilation

2.5.3 Dilated convolution

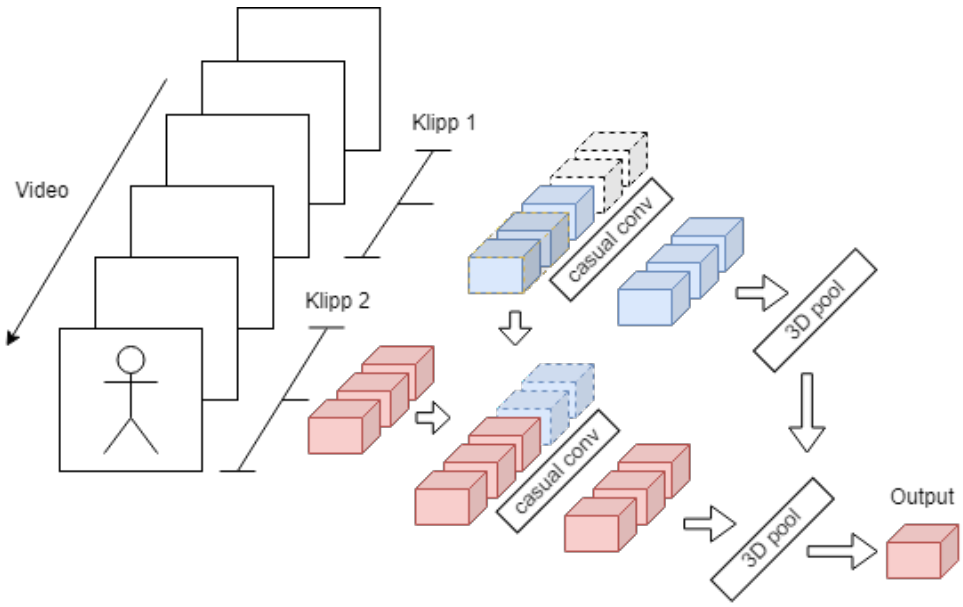
Dilated convolution går ut på å endre hvordan convolution filteret skal brukes. Filteret vil se likt ut, men i stede for å direkte bruke filteret på hosliggende datapunkter, hopper man forbi datapunkter med en dilation faktor. I figuren 2.6 viser det hvordan man hopper over datapunkter ved å skalere dilation over hvert lag. Hvis man ikke hadde bruk dilation vil synsfeltet til convolution skalere linjert. På den andre siden ved bruk av skalerende dilation for eksempel mengden {1, 2, 4, 8} vil man kunne øke synsfeltet eksponentielt. Dermed kan man hente ut globale sammenhenger i dataen, uten å drastisk måtte øke modellens kompleksitet. Slik convolution er for eksempel brukt i Vesal Sulaiman, 2019 for å forbedre segmenterings resultater ved 3d convolution og i Yangdong He, 2019 hvor det brukes for å konkurrere med rnn nettverk for sekvensiell data.

2.5.4 Casual convolution

For å bruke en stream-buffer, som jeg kommer tilbake til senere, må man bruke casual convolution. Denne convolution metoden sørger for at fremtidig data, for eksempel i tidsserier, ikke påvirker gammel data. I figur 2.6 kan man se at convolution lagene danner en pyramide. I modeller som skal klassifisere en 3d modell vil ikke dette gjøre noe forskjell, fordi alle 3 dimensjonene er romlige, for eksempel i Vesal Sulaiman, 2019. På den andre siden vil ikke dette gi mening hvis man skal bruke modellen på predikering av en tids basert sekvens (van den Oord mfl., 2016). Hvis man skal bruke casual convolution for høyere dimensjonal data kan man bruke maskert convolution. Dette er ikke nødvendig hvis man bare skal bruke de i en dimensjon, der man kan flytte resultatet av vanlig convolution med noen steg, basert på padding størrelsen for å opprettholde original datastørrelse (van den Oord mfl., 2016).

2.5.5 Stream Buffer

Hvis en modell skal evaluere et langt klipp, i vårt tilfelle opptil 200 bilder, vil dette fort gå over minnekapasiteten. Ved å dele opp klippet i mindre deler, motvirkes minneproblemene. Utfordringen er hvordan modellen skal ta hensyn til alle klippene ved klassifisering. En naiv modell kan for eksempel kjøre hvert klipp gjennom modellen og ta gjennomsnittet av klippene. Den naive modellen har ulempen med at den ikke ser på tverr-klipps sammenhenger ved klassifisering. En bedre metode er bruk av en stream-buffer. Stream-buffer implementasjonen er inspirert av Kondratyuk mfl., 2021. En stream-buffer fungerer ved å bruke causal convolution for å lagre tidligere activation i de siste delene av forrige klipp. Casual convolution vil føre til at all tidligere kontekst i klippet legges sammen i siste bit av klippet. Ved å ta et par datapunkter i slutten av forrige klipp og legge dem til foran neste klipp før gjennomkjøring vil all kontekst bevege seg gjennom klippene. Forskjellen med metoden og andre rekursive modeller vil være hvor vi henter ut resultat sekvensen. Figur 2.7 viser hvordan en stream-buffer kan se ut. Modellen vil ligne en mange-til-mange rekursiv modell. Etter alle klipp er kjørt gjennom modellen, vil tredimensjonal-kumulativ-gjennomsnitt pooling brukes for å legge klippene sammen og hente ut resultatet.



Figur 2.7: Stream-buffer med 3 bilder pr. klipp. 2 bilder fra forrige klipp blir lagt til neste klipp før convolution. 3D pooling blir brukt for å få endelig resultat

3. Metode

I dette kapittelet skal jeg gå gjennom hvorfor jeg gjorde de valgene jeg gjorde, og dermed også hvilket valg jeg måtte revurdere gjennom prosessen. Alle valgene jeg gjorde var rettet mot best mulig oppfyllelse av problemstillingen. I tillegg er valgene som ble gjort bygd opp med teoretisk bakgrunn fra teori kapittelet. Enkelte valg som ble gjort i startfasen måtte endres gjennom prosessen, der ny kunnskap oppkom. Slike valg vil bli spesifisert.

3.1 Data

Først må jeg gå gjennom hvilket datasett som skal klassifiseres. Problemstillingen spesifiserer at datasettet må være naturlige øyebevegelser. Dataen må være videoer av øynene, gjerne også tatt med vr / vr-lignende kameravinkler. Dataen må omhandle begge øynene. Ut ifra kriteriene fra problemstillingen analyserte jeg en rekke datasett, som vist i tabell 3.1

Ut ifra tabell 3.1 fant jeg ingen datasett som matchet aller kriteriene. Dermed måtte jeg revurdere problemstillingen, og fjerne kriteriene om kravet om to øyner. Dette vil ha en effekt på problemstillingen i hvilket øyebevegelser jeg må ta hensyn til. Øyebevegelsene går jeg dypere inn på i kapittelet som omhandler dem. Videre skal jeg gå dypere inn på hva som differensierer de mest lovende datasettene, og hvordan jeg gikk frem i å velge det resulterende datasettet.

3.1.1 Open EDS

Dette er datasettet som jeg valgte for oppgaven. Datasettet er delt opp i 4 mindre deler. Del 1 er 12759 bilder med annotering for nøkkel punkter i

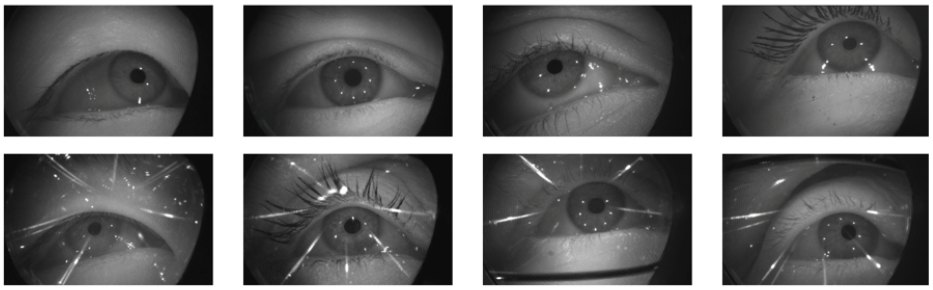
Datasett	Video	Naturlig	vr-ligndende	begge øynene
Open EDS	x	x	x	
Gaze-in-wild	x	x	x	x
SynthesEyes	x			
Dr(eye)VE		x		
EEGEyeNet				
Provo Corpus				

Tabell 3.1: Forskjellige datasett, og om de matcher kriteriene

øyet. Del 2 består av 252690 ikke annoterte øyebilder. 3) 91200 tilfeldige valgte bildesekvenser av 1.5 sekund varighet. 4) 143 par med høyre og venstre punkt data for øyet, hentet ved bruk av hornhinnetopografi (Garbin mfl., 2019). Del 3 av datasettet stemte overens med 3 av 4 krav fra problemstillingen. Bildene i denne delen er tatt med ir-kamera via stasjonære briller. Bevegelsene øyet gjør er naturlige, ettersom opptakene blir tatt ved deltagere som ser på tilfeldige valgte bilder fra imagenet datasettet. Selve datasettet ble lagd for å fremme vr-teknologi, og passer dermed også bra opp mot problemstillingen.

3.1.2 Gaze-in-wild

Gaze-in-wild datasettet var også en god kandidat, men ble ikke valgt. Datasettet inneholder øye og hode rotasjons hastigheter, infrarøde bilder av øynene og scenebilder som ble brukt (Kothari mfl., 2020). Datasettet oppfylte 4 av 4 krav for problemstillingen. Datasettet var større enn Open eds, med over 2 TB bildedata. Dataen kommer i 2 mp4 format videoer pr øye, der hver video ikke er delt opp i brukbare videoklipp. For å bruke dette datasettet måtte man kutte og omstokke videoene. Jeg bestemte at dette var unødvendig, ettersom størrelsen av datasettet var urimelig, i forhold til oppgavens omfang. Størrelsen på open eds og data som kom ferdig behandlet var grunnene til at jeg valgte det.



Figur 3.1: Eksempler på ir bilder fra Opend EDS datasettet

3.2 Øyebevegelser

Øyebevegelsene som jeg først tenkte var rettmessige for oppgaven, som nevnt i teori biten, var ikke alle like gunstige. Datasettet jeg bruker har kun bildet av et av øynene om gangen, ved analyse av vegensbevegelser vil dette være et problem. For klassifisering av vergens bevegelser må man se på forskjellen mellom begge øynene, som ikke blir mulig. vestibulo-okulære bevegelser vil ikke kunnes skilles fra sakkader. Vor bevegelser

kan bare skilles hvis man også har hodets bevegelse ved analyse. Begge bevegelsene kan analyseres i Gaze-in-wild datasettet, men ettersom problemstillingen påpekte at modellen skal være et konseptbevis og ikke et ferdig produkt, bestemte jeg å simplifisere ved å se over bevegelsene. Post sakkadiske overskridelser ser jeg også over, ettersom modellen vil se på fast lengde bildesekvenser. Bildesekvensene må ha en lengde som kan analysere fikseringer, jevn forfølgelse og sakkader, som blir vanskelig hvis vi også skal ta hensyn til små post sakkadiske bevegelser. Dette er også grunnen til at fikserings bevegelser ikke blir tatt hensyn til.

3.3 Modell implementasjon

Under gjennomføring av prosjektet hadde jeg ikke tilgang til prosesserings kraft før mot slutten av prosjektet. Dette førte til en rekke estimasjoner over modellens arkitektur. De første iterasjonene av modellen var basert på to dimensional convolution, og testet på datasettet MNIST. Datasettet er ofte brukt for å teste bilde analyse modeller. Modellene som ble foreslått var en kontrastiv lærings modell og en auto-enkoder.

Øyebevegelses data fra Open EDS består av 3 dimensjonal data. De to første dimensjonene er romlige dimensjoner (bilder av øyet) den 3. dimensjonen er en tids dimensjon. En enklere løsning av problemstillingen, kunne vært å bruke en api for å hente ut nødvendig informasjon ved datasettet, for eksempel vinkelhastighet på pupillen. Dette vil redusere dataen til 2. dimensjoner. Første dimensjon vil være featurer data og andre dimensjonen som tids dimensjon. Unsupervised læring på slik 2 dimensjonal data er generelt en lettere oppgave. Problemet med en slik løsning er at, ved uthenting av featurenes vil ikke modellen være en ende-til-ende-modell. Dermed valgte jeg å bruke modeller som kan håndtere 3. dimensjonal data.

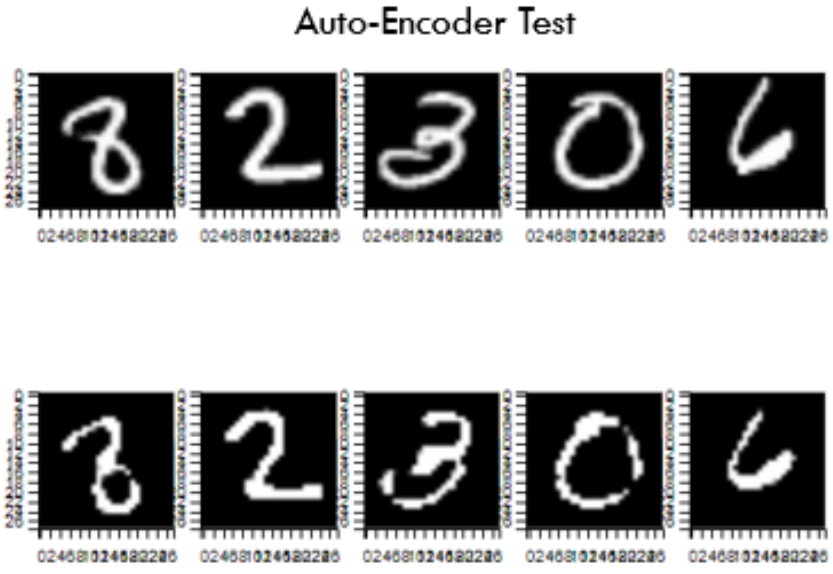
Jeg fant ikke mye forskning på slike 3. dimensjonale unsupervised modeller, og måtte derfor teste flere forskjellige modeller føre jeg kom fram til den endelige modellen. 3. dimensjonale modeller for video klassifikasjon er mer kjent. For eksempel Movinet (Kondratyuk mfl., 2021), som er en 3d cnn basert modell for å klassifisere video. Movinet kan bruke data størrelser som ligner datastørrelsen fra øyebevegelse dataen. Modellene jeg utviklet tar inspirasjon fra dette nettverk.

3.3.1 Auto enkoder 1. iterasjon

Auto enkoder modellen har en enkel arkitektur, der enkoderen ligner arkitekturen i den kotrastive modellen, og dekoderen har lik arkitekturen bare

snudd. For opp og ned sampling bruker jeg avragepool og transposed convolution. Denne modellen ble også kjørt på MNIST datasettet.

Denne gjennomkjøringen ga gode resultater. I figur 3.2 kan du se hvordan auto endkoderen reproduserer bildene som blir sent gjennom dem. Taps funksjon jeg valgte for gjennomkjøringen var ikke gunstige, ettersom den gjorde at resultatet ble en binær representasjon. Dermed endret jeg denne til senere kjøringer. Dette resultatet er en indikasjon på at modellen er god, vi kan få en bedre indikasjon hvis vi henter ut enkoderen fra modellen, og kjører bare med denne. Dette må vi gjøre ettersom det er enkoderen vi bryr oss om for transfer læring i sluttproduktet. Det er vanskelig å visualisere en enkoder i en auto enkoder, fordi denne vil vise abstrakte sammenhenger modellen har funnet. Dette er spesielt synlig ved bruk av en linjer flaske hals. Denne modellen bruker en linjer flaskehals, men viste fortsatt lærte sammenhenger i datasettet. For å teste om enkoderen gir gode resultater, bruker jeg en SVM (support vector machine). SVM kan vise linjer og ikke linjere sammenhenger. Ved trening av en SVM modell på enkoderen, fikk jeg en klassifiserings treffsikkerhet på 97%. Treffsikkerheten er god nok for at jeg velger å gå videre med modellen.



Figur 3.2: Bilder fra MNIST datasettet, før og etter auto-enkoderen

3.3.2 Kontrastiv læring modell 1. iterasjon

Før jeg prøvde modellen på faktisk datasett, prøvde jeg en konsept-modell. Denne konsept modellen hadde en simpel arkitektur, bygd opp på 4 vanlige convolution lag og 2 fully-connected linjere lag. Selve arkitekturen er ikke så viktig, ettersom dette bare er en konsept-modell. I tillegg brukte jeg tilfeldig datatransformasjon for å oppfylle kravene for en slik modell. Modellen ble trent på MNIS datasettet.

Resultatet av kjøringene var dårlige. I Ting Chen, 2020 beskriver de at en kontrastiv modell trenger god datatransformasjon for å kunne trenes. Modellen blir trent på datasettet mnist, og var derfor holdt tilbake på grunn av dette. I tillegg, I Ting Chen, 2020 forklare de at større modeller og lengre trenings periode påvirker modellen positivt. Ved testing med en SVM på resultatet, fikk jeg en treffsikkerhet på 0.88%. Ettersom jeg ikke forventet gode resultater, vurderte jeg derfor at modellen burde testes på Open EDS datasettet.

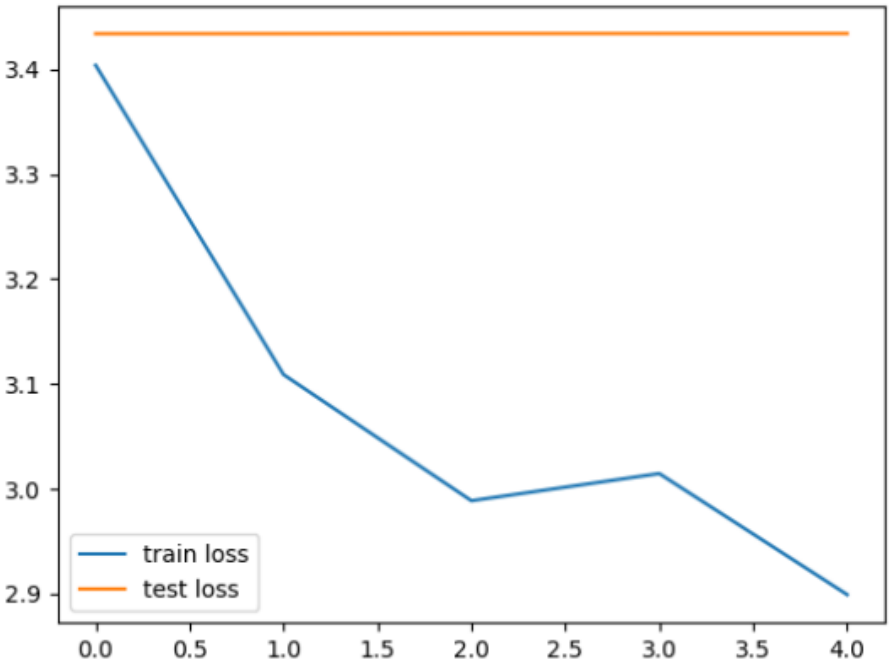
Etter testing av de første modellene, kom jeg frem til at jeg trenger ikke linjere klynge metoder. Ideene først var å bruke vanlig k-means, men ettersom begge unsupervised modellen ikke ga linjere resultater måtte jeg re-evaluere hvilken metode som burde brukes. Etter gjennomgang valgte jeg å bruke en kernal PCA isteden. De neste modellene kjøres alle på Open EDS datasett. De neste modellene bruker 3d convolution.

3.3.3 Kontrastiv læring 2. iterasjon

Utfordringene med å bytte til 3d convolution, var svært store minne problemer. Ettersom ved backpropegation trenger modellen å lagre activations, ville modellen fort fylle opp gpu minne. Alle modellene ble trent på Nvidia 4090, men selv med 24 gb minne, var dette fortsatt et problem. Fellestrekk ved de andre iterasjonene var like enkoder, og data input størrelse på 256 x 256 x 60. Datastørrelsen måtte reduseres for å holde modellen innenfor. Arkitekturen jeg brukte var hentet fra Movinet, men redusert ettersom modellen måtte kjøre to gjennomkjøringer pr backpropegation. Dette er svakheten ved slike unsupervised modeller. Auto enkodere har samme problemet ettersom de trenger en enkoder og en dekoder. For å få samme resultater som lignende supervised modeller, trenger man dobbelt så mye minne. Jeg valgte å implementere stream-buffer for å minnke minnebruk.

Modellen bruker ikke fullt 3x3x3 convolution filter, men benytter 2+1 convolution (spatiotemporal convolution). Dette gjøres ved å splitte opp filteret i 3x3x1 og 1x1x3 filter, og ha egne convolution lag for begge. Et

slik 2+1 dim convolution lag reduserer trenings varighet (Kondratyuk mfl., 2021). I Du Tran, 2018 ser de at spatiotemporal convolution gir bedre treffsikkerhet og lavere risiko for overfitting, i motsetning til ekvivalente 3d convolution nettverk.



Figur 3.3: Trenings og valideringstap for kontrastiv læring modell nr. 2

Resultatene for den kontrastive lærings baserte modellen, var ikke gode. Modellen greide ikke trene. I Ting Chen, 2020 sier de at slike modeller trenger store batch størrelser, for å trenes. Ved trening var modellen hold tilbake av minne kostnader, og dermed måtte modellen trenes på lav batch størrelse. Dette kan være en av grunnene til at modellen ikke ville trene. Oppgavens høy dimensjonale datastørrelse kan også være en grunn for problemer med denne metoden. En slik metode er kritisk avhengig av god data behandling. Tilfeldige translasjon og rotasjon i tillegg til Gaussian støy, var metodene jeg brukte for tilfeldige databehandling. Dette kunne også vært et problem ved trening. Jeg har ikke sett kontrastive taps modeller brukt for slike 3 dimensjonale bruksområder. Med hensyn på resultatene valgte jeg ikke teste en kontrastiv modell videre i prosessen.

3.3.4 Auto enkoder 2. iterasjon

Andre generasjon auto-enkoder brukte bare convolution baserte lag. Her fjernet jeg stream buffering, causal convolution og flaskehalsen. Ved å ikke bruke slike lag vil vanlig convolution fører til lokale, linjere egenskaper i enkoder resultatet. Slike lokale egenskaper er lettere å rekonstruere ved bruk av færre convolution lag i dekoderen. Dette er av samme grunner som førte til bruk av kernel PCA. Grunnene bak dette gikk jeg gjennom i kapittelet ”Auto enkoder 1. iterasjon”.

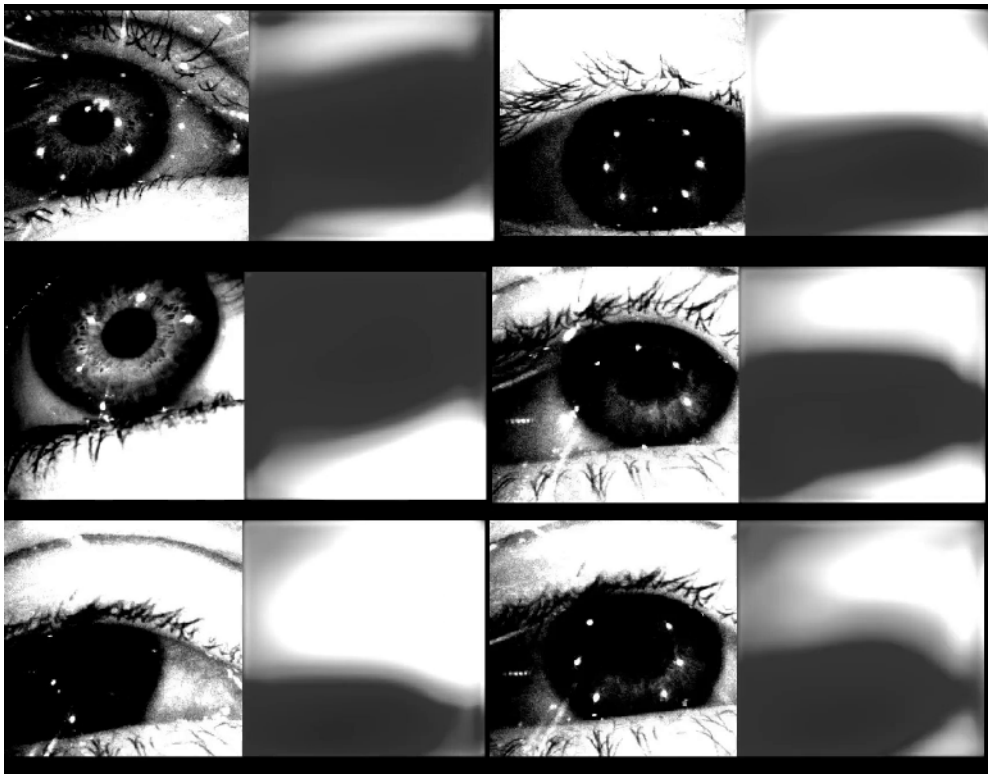
Oppgaven å rekonstruere en video er svært vanskelig. En video på 60 bilder, der bildene er 256 x 256 piksler, må modellen generere nesten 4 millioner piksler. For en slik oppgave trenger modellen en stor datamengde gjennom flaskehalsen. Hvis flaskehalsen er for stor, vil ikke klyngealgoritmer som k-means fungere. Dermed må flaskehalsstørrelsen balanseres mot reproduksjons evne, klynge evne og under/over fitting av flaskehalsen. I tillegg bruker jeg ingen form for downsampling / redusering i den temporale dimensjonen. Valget ble gjort ettersom video klassifiserings metoder som movinet ikke gjør redusering i tids dimensjonen før siste steg ved evaluering. Jeg trodde først upsampel-ingen, nødvendig ved reduksjon, ikke ville ødelegge de temporale sammenhengene. Under senere testing oppdaget jeg at det ikke stemmer.

Modellen ga dårlige resultater, pga. redusert filtermengden i convolution lagene. Reduksjonen var nødvendig for å ikke overskride minne begrensninger. Flaskehalsen hadde for mye data, og ville dermed gitt dårliger resultater ved klynging.

3.3.5 Auto enkoder 3. iterasjon

Tredje iterasjon brukte enkoder med stream-buffer, med en flaskehals. Flaskehalsen bestod av en dilated convolution del og en linjer del. Dilated convolution delen bestod av 3 lag med dilation fra mengden 1,2,4. Den linjere delen var 3 linjere lag. Dekoderen brukte ikke en stream-buffer, ettersom jeg ikke vet om det er mulig. Denne modellen kunne ha flere filter, ettersom stream-bufferen reduserte minnebehovet for modellen. Dekoderen måtte derfor hver mindre enn enkoderen for å holde modellen innenfor minne restriksjonen.

Dekoderen bruker nærmeste-nabo upsampeling. Vanligvis bruker for eksempel u-net transposed convolution for upsampeling (Ronneberger mfl., 2015), men nærmeste-nabo upsampeling bruker ikke lærte parametere, og dermed mindre minne. Høyere minnebruk på normal convolution ga bedre resultater enn transposed convolution.

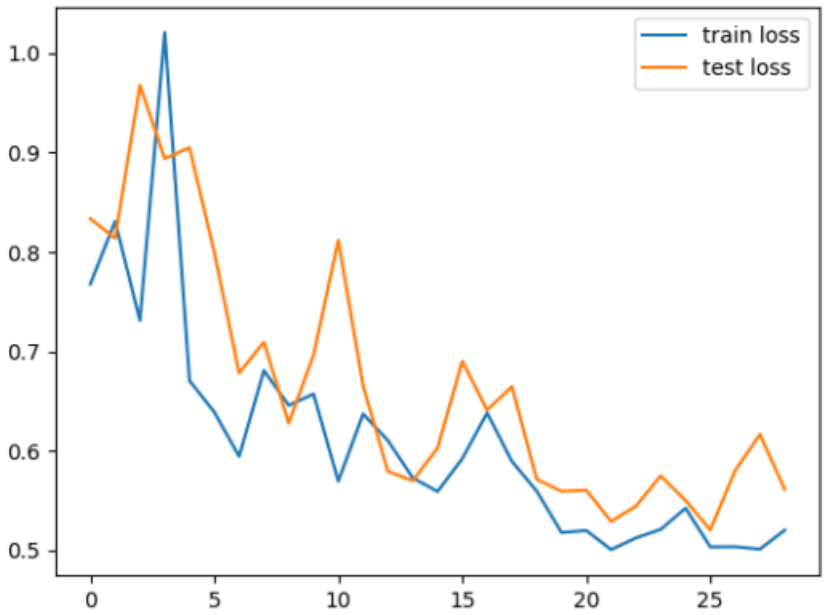


Figur 3.4: Tilfeldig valgte bilder fra auto-enkoder gjennomkjøring

Figuren 3.4 representerer bilder før og etter auto enkoderen. Det er ingen pupille, og alle bevegelsene er trege. Resultatet er ikke bra nok for å gjøre en god klassifikasjon av slike øyebevegelser. Problemet er enten at dekoderen, flaskehalsen eller enkoderene underfitter. I figur 3.5 kan man se tegn på underfitting. Det kan også være et problem at datastørrelsen gjennom flaskehalsen er for liten. Modellen kan ikke bli dypere pga. minne begrensninger, og økning av flaskehalsen størrelsen vil gjøre klynge resultatet dårligere.

3.3.6 Auto enkoder 4. iterasjon

For å motvirke minne problemet, valgte jeg å redusere data størrelsen. Problemstillingen handler om klassifisering av øyebevegelser. Av øyebevegelsene som skal klassifiseres har sakkader kortest varighet. Sakkader har vanligvis en varighet på under 50 ms. Modellen som er foreslått vil kunne klassifisere en øyebevegelse pr datapunkt. Teoretisk kan fikseringer, sakkader og jevn forfølgelse bevegelse klassifiseres med et varighets spenn på 50 ms. Modellen trenger ikke fulle video sekvenser på 1.5 sekunder, for å kunne oppnå klassifiseringen. Dermed reduserte jeg datasekvensene til



Figur 3.5: Tap for auto-enkoder iterasjon 3.

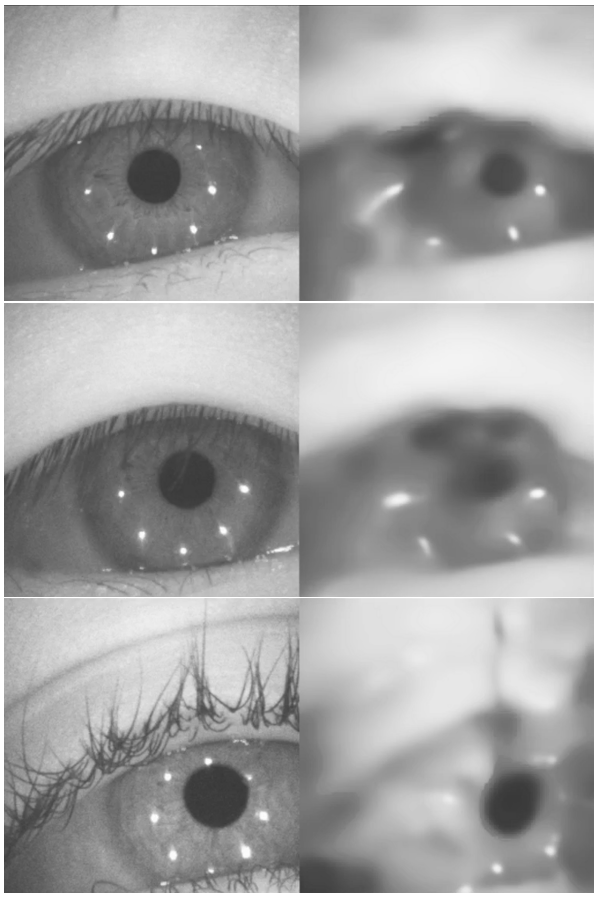
6 bilder pr sekvens, som er en god balanse mellom varighet og minnebruk.

Den reduserte datamengden, gjør oppgaven å reprodusere den originale videoen lettere. Modellen kunne hente ut flere viktige trekk ved datasettet, og redusere datamengden som måtte gjennom flaskehalsen. Den reduserte flaskehalsen vil forhåpentligvis forbedre resultatet ved bruk av klynging av data. I tillegg kunne jeg legge til flere lag i enkoderen og dekoderen, ettersom minne bruken var redusert. Convolution bruker ikke mer parametere ved større datamengde, men ved backpropogation vil pytorch lagre actiavations. I pytorch eksisterer det checkpoints, for å redusere minnebruk ved lagring av actiavtions mot datakraft ved re-kalkulering. Denne metoden kom jeg over etter gjennomføring av prosjektet, og er ikke brukt i den endelige modellen.

causal convolution og stream buffer var ikke nødvendig i den nye implementasjonen. Metodene er gode for lange data sekvenser. Den nye redu-

serte dataen vil ikke trenge metodene, for å holde minne nede. I den nye modellen bruker jeg vanlig 3. dimensjonal convolution, med downsampling i alle dimensjoner. Ved causal convolution er det viktig å opprettholde hele data sekvensen gjennom modellen. Den nye modellen vil da bruke mindre minne, ettersom den kan redusere lengden av datasekvensen gjennom modellen. Den reduserte datamengden vil føre til færre lagrede actiavtions for backpropagation.

Modellen ga de beste resultatene av alle modeller jeg prøvde. I figur 3.6 har det genererte bildet tatt med pupillen og bevegelser. Problemstillingen sier at modellen burde ta hensyn til et helhetlig bilde. Ved analyse av de resulterende videoene fra modellen, ser vi at det ikke bare er pupille bevegelser og diameter. Klassifiseringsmodellen vil da ta hensyn til flere trekk, enn tradisjonelle øyebevegelse algoritmer.



Figur 3.6: Resultater ved auto-enkoder iterasjon 4

3.3.7 Auto enkoder 5. iterasjon

Modell 5. prøver å optimalisere parametere fra modell 4. Den største endringen var å redusere flaskehals størrelsen. Mindre flaskehals størrelse vil gi bedre resultater ved klynging senere. I tillegg flyttet jeg parametere fra enkoderen til dekoderen. Dette vil teste om dekoderen hadde et underfitting problem i modell 4.

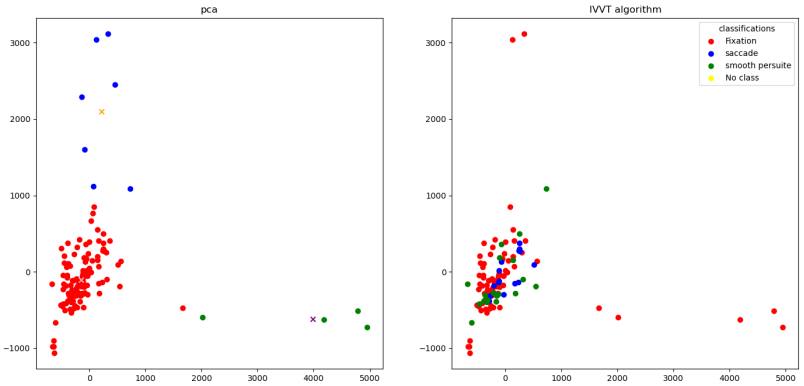
Modellen ga dårligere resultater enn modell 4. Ved testing ble det kjørt tre forskjellige varianter, der alle hadde litt forskjellige parameter balanse mellom enkoder og dekoder. Alle kjøringene brukte den reduserte flaskehalsen. Resulterende videoer er vedlagt. Modeller med redusert flaskehals, relativt til modell 4, ga ikke optimale resultater. Ved klyngetesting vil jeg dermed bruke modell 4 som slutt modell.

4. Resultater

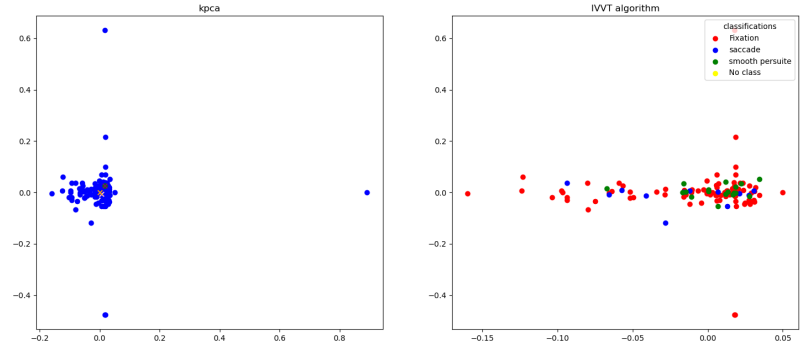
Enkoderen ble hentet fra auto enkoderen fra iterasjon 4. Flaksehalsen blir også hentet fra denne modell iterasjonen. Det blir prøvd med både pca og kernel pca. Deretter blir kmeans brukt for klynging. Det er totalt tre klyngesentra brukt. De representerer grupperingene sakkader, fikseringer og jevn forfølgelse bevegelse. Grupperingene er basert på de enkleste skille-trekkene mellom data punktene. Enkoderen reduserer datapunktene fra størrelsen 256x256x6 = 393216 til 4096 trekk. Datapunktene blir dimensjon redusert med 9600%. Forhåpentligvis vil dette gjøre det mulig for k-menes å klynge datapunktene.

Resultatene vises i graf form, ettersom dette gjør det lettere å se klynge og sammenhenger. Modellene blir dimensjons redusert på forskjellige måter. Det dimensjons reduserte resultatet blir deretter klynget med k-means. Klyngene har forskjellige farger i grafene. Hver figur har også IVVT resultatene for sammenligning.

Grafene er to dimensjonale, men dimensjons reduksjonen reduserer til 40 dimensjoner. Dette vil føre til dårlig visuell representasjon, men bedre antagelser. IVVT algoritme tar bare hensyn til videoer der den får en pupille posisjons måling. Enkelte punkter vil ikke vises i IVVT resultatene.



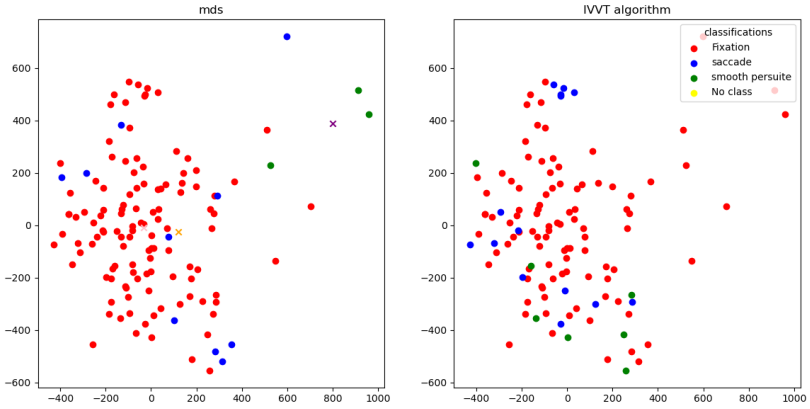
Figur 4.1: modele 4. med PCA dim reduksjon. Venstre er Kmeans resultat. Høyre er IVVT resultat.



Figur 4.2: modele 4. med kernel PCA dim reduksjon. Venstre er Kmeans resultat, Høyre er IVVT resultat.

I tillegg til pca i figur 4.1 og kernel pca i figur 4.2 dim reduksjon, prøvde jeg også tsne figur 4.3, isomap og mds dim reduksjon. Isomap og mds resultater er vedlagt.

Tabell 4.1 er fordelingene av de tre forskjellige klassene, i stigende rekkefølge. IVVT resultatet beskriver hvordan en fordeling burde se ut. Fordelingene er ikke representasjoner av fiksering/sakkader/gjevn forfølgelses bevegelse. Fordelingene må brukes i sammenheng med grafene, for å se om gruppene overlapper. En god modell vil ha sammenlignbare fordelinger, og likheter i grafene.



Figur 4.3: modele 4. med MDS dim reduksjon. Venstre er Kmeans resultat, Høyre er IVVT resultat.

Metode/fordeling	low	mid	high
IVVT	0.0598	0.0940	0.8461
PCA	0.0156	0.4453	0.5390
KPCA	0.0078	0.0078	0.9843
TSNE	0.3203	0.3203	0.3593
ISOMAP	0.0234	0.3751	0.6015
MDS	0.0234	0.0781	0.8984

Tabell 4.1: Fordelinger av de 3 grupperingene

5. Diskusjon

5.1 resultater

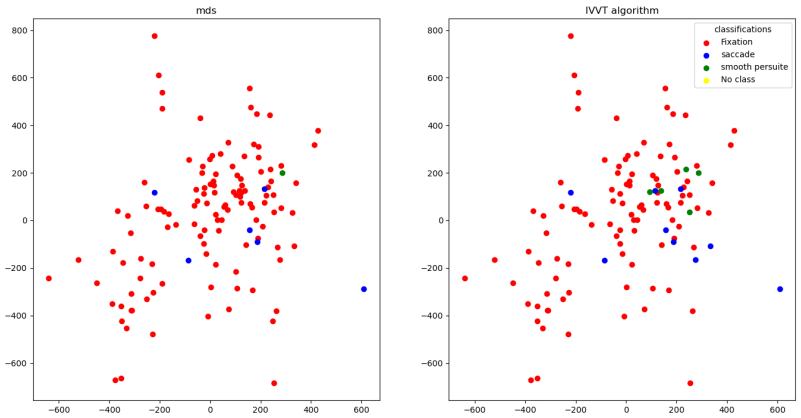
Resultatene fra gjennomkjøringene svarer ikke på problemstillingen. De beste modellene, der forholdene mellom grupperingene er best, er det ingen overlapp med IVVT algoritme. MDS algoritmen og kernel PCA ga de beste resultatene. Resultatene fra metodene overlappet med forventet øyebevegelses fordelinger. Figur 4.1 viser at fikserings fordelingen mellom MDS og IVVT var 0.8984 og 0.8461. Fordelingen mellom middels gruppering og lavest gruppering var også forholdsvis like. Dette kan antyde et godt klynge resultat, men dette er ikke korrekt. Grupperingene ble gruppert etter andre trekk enn det som brukes for klassifisering av øyebevegelser. Figur **??** viser at det ikke er sammenheng mellom sakkader, jevn forfølgelse bevegelse og grupperinger av kernal PCA metoden. Ettersom modellen ikke klassifiserer øyebevegelser, oppfyller dette ikke problemstillingen.

5.2 Mulige problemer

Oppgaven er vanskelig, spesielt for modeller basert på unsupervised læring. Unsupervised lærings modeller vil trekke ut de tydeligste sammenhengene ved klynging. Modellen kan mene at øyets utseende, eller om øyene har briller eller ikke, er klarere sammenhenger enn øyebevegelser. Dermed vil bruk av dimensjons reduksjon og klynging klassifisere slike sammenhenger over de ønskede sammenhengene.

Den sammenslåtte dataen fra flaskehals og enkoder er ikke et problem, men dimensjons reduksjon og k-means er problemet. Auto-enkoderen greide å rekonstruere originalbildene, og har dermed nødvending sammenhenger i enkoder produktet. I figur 5.1 tester jeg dette med å bruke en SVM (support vector machine) for klassifisering. SVM er en supervised metode, som tvinger modellen til å bruke de sammenhengen som blir spesifisert. Input data vil være enkoder resultatet fra modell 4, og annoteringene vil være IVVT resultatet. I figur 5.1 er det overlapp mellom SVM klassifiseringer og IVVT klassifiseringer.

Auto enkoder flaskehalsen er ikke linjer og dermed vanskelig å gruppere. Ikke linjer data er generelt vanskelig å gruppere. Ikke linjere klynge



Figur 5.1: modele 4. med MDS dim reduksjon. Venstre er SVM resultat, Høyre er IVVT resultat.

oppgaver, med stor data dimensjon, trenger mer avanserte metoder for klynging. Min implementasjon brukte forskjellige dimisjons reduksjon metoder, men bare vanlig k-means. Bruk av andre klyngemetoder kan gi bedre resultater enn det jeg fikk. Valget av k-means ble gjort på et grunnlag av å redusere implementasjons komplexiteten. I stedet valgte jeg å bruke mer tid på testing av dimensjons reduksjons metoder.

Bruk av en vanlig auto enkoder, kan være mindre gunstig for unsupervised klynging. Ettersom auto-enkodere krever reproduksjon av original data, vil ikke ønsket data være nødvendig i flaskehalsen. Bruk av bedre taps funksjoner kan motvirke dette problemet. For eksempel ved å ta mer hensyn til differansene mellom bilder i sekvensen, kan vi lære modellen å produsere bevegelser, og ikke statiske trekk. En slik modell vil trenge færre data dimensjoner i flaskehalsen, og dataen som går gjennom flaskehalsen vil være mer relevant for klassifisering.

Minnerestriksjoner kan føre til dårligere resultater. Ved å bruke en maskin med større minne kapasitet, kan man bruke flere input bilder pr sekvens eller større modeller. Som sagt tidligere i rapporten er det en rekke minne optimaliserings teknikker som burde bli implementert. For eksempel torch checkpoints og redusert bilde størrelse. Flere bilder pr sekvens vil gi modellen mer informasjon for å differensiere jevn forflyttelse bevegelse og fikseringer. I tillegg kan man bruke dypere enkoder – dekoder modeller, som gir bedre resultater i flaskehals.

Flaskehalsen jeg implementerte kan gjøre klynging vanskeligere. De linjere lagene vil gjøre flaskehals produktet mer abstrakt/ikke linjert. Flaske-

halsen kan også være unødvendig. Dilation flaskehalsen kan være unødvendig, hvis modellen allerede har global kontext fra mengden lag. På samme måte kan linjere flaskehalsen gi dårligere resultater, ettersom det er vanskeligere å reprodusere i dekoder. Flaskehalsen bruker også store mengder parametere. Spesielt den linjere flaskehalsen, som ikke er basert på convolution. Flytting av minnebruken fra flaskehalsen til enkoder/dekoder, kan gi bedre resultater.

IVVT algoritmen og pupille posisjons uthentings metoden var dårlige. Problemstillingen spesifiserer at dette ikke skal være en sluttmodell, men et bevis på konseptet. Dermed ble mindre tid allokert til kalibrering av grense verdier for IVVT. Metoden for uthenting av pupille posisjon var kontrast basert, og ikke kalibrert. Treffsikkerheten av metoden vil være mindre gunstig, men vil kunne gi en antagelse av de riktige grupperingene.

Hyper-parametere i auto-enkoderen var ikke optimale. Ved mer testing kunne modellen optimaliseres bedre med hyper-parametere. Parameteren jeg valgte var basert på movinet (Kondratyuk mfl., 2021), og generell testing. Ved bedre testing kunne for eksempel filter størrelsene og antall kanaler optimaliseres for et bedre enderesultat.

5.3 Prosess

Prosjektet bestod av en kombinasjon av prøving, feiling og teori lesing. Tre dimensjonale auto enkodere og klynging var vanskelig å finne teori rundt. Artiklene jeg fant var ofte dårlige eller ikke stemte med kravene. Modellene ble fremstilt ved antagelser byggd opp av teorien jeg hadde lest. Tidlig i prosessen var modellene mindre gode, ettersom jeg ikke hadde god nok teoretisk forståelse. Gjennom prosessen var målet å bygge opp kunnskaper, og dermed finne ut hva som var mulig å lage, og hvordan dette skulle lages. Problemstillingen måtte derfor endres gjennom prosjektet, ettersom hva som var mulig å gjennomføre. Endringer av problemstilling ble tatt opp og godkjent med veileder.

Prøving og feiling av forskjellige antagelser var viktig under prosjektet. Problemstillingen startet åpen og uten grundige kravs spesifikasjoner. Under metode kapittelet går jeg gjennom hoved iterasjonene av modellene jeg testet. I tillegg til modell iterasjonene ble det også testet en mengde mindre komponenter, som for eksempel forskjellige actiavions, taps funksjoner, residual-blokker, normalisering (lag og bach), skip koblings typer (concat/addition), upsampling metoder (transposed conv / interpolasjon), downsampling (avg pool, max pool), regularization (L1, L2, Dropout), osv. Testing var viktig for prosessen, ettersom lignende modeller ikke er godt

dokumentert eller testet.

Testing av forskjellige modeller og metoder var tidskrevende. De tidligere modellen tok ca. 2-3 timer å trene. Jeg reduserte trenings tiden for å finne den mest effektive metoden. De senere modellene tok ca. 5-6 timer å trene. Arbeidsprosessen ville da være lesing/testing, optimalisering, og deretter testing/trening av ny modell. En limiterende faktor for prosjektet var mengden maskintid jeg hadde på NTNU maskiner. Jeg fikk ikke tilgang til maskin før starten av april. Gjennom april testet og utviklet jeg modell iterasjon 2 og utover. Maskinen var også ofte opptatt store deler av dagen, som førte til færre gjennomførte tester.

Forskjellige modeller og strategier ble foreslått og diskutert med veileder. Dette hjalp med fremstilling av en forbedret problemstilling og reduksjon av oppgavens omfang. Teori og implementasjon var også diskutert med veileder. Diskusjonene var ikke hjelpsomme, ettersom veileder ikke hadde spesiell kunnskap innen 3d auto-enkodere eller unsupervised læring. Ved bedre veiledning og teori, kunne oppgaven siktes mer mot et ferdig produkt. Oppgaven er ikke relevant for folk som ikke er spesielt interesserte i dyp læring.

5.3.1 Videre arbeid

Problemstillingen burde gjøres enklere ved videre arbeid. Modell 4. har vist gode resultater ved enkoding, men klyngene ga ikke ønsket resultat. Ved å ikke bruke en full ende-til-ende metode kan modellen få bedre resultater. Ved bruk av en supervised featuere uthenter, for eksempel for pupille posisjon/hastighet, vil klynge resultatene bli bedre. Ved å hente ut relevante sammenhenger i dataen, kan vi redusere total data dimensjon drastisk før bruk av unsupervised klynge algoritme. Klynge algoritmer gir bedre resultater ved lav dimensjon data, og data som bare har relevante trekk. I tillegg eksisterer det flere datasett som spesialiserer seg på slik data, for eksempel Provo Corpus. Testing av forskjellige unsupervised klynge metoder på slik data ville vært svært relevant for problemstillingen.

Bruk av supervised læring med tre dimensjonal enkoder kunne også vært interessant. Her vil modellen fortsatt være en ende til ende modell. Kjøretiden ved en slik modell ville vært mindre enn lignende to part modeller. Implementering av en forenklet movienet (Kondratyuk mfl., 2021) kunne gitt et godt grunnlag for en slik modell. Her er det viktig at det brukes et godt annotert datasett, ettersom ved supervised læring vil modellen etterligne annoteringen. En modell basert på dette, vil ha gode muligheter for å slå alle eksisterende algoritmer. Jeg fant ikke en slik implementasjon ved leting etter teori. Hvis man vil gjøre oppgaven mer interessant, kun-

ne man brukt teknikker fra efficent-net, for å gjøre modellen kjappere. På den andre siden kan en implementasjon av en tre dimensjonal supervised vison transformer (Jean Lahoud, 2022) være interessant, hvis best mulig treffsikkerhet er oppgavens mål.

En ende-til-ende modell med unsupervised læring er mulig, men da må mer avanserte metoder brukes. En Disentangling Variational Autoencoder (Irina Higgins, 2016), uten å gå inn på hvordan en slik auto enkoder fungerer, kan gi bedre resultater. Kontrastive metoder er også lovende, selv om min implementasjon ikke fungerte. Modellen jeg foreslo hadde blant annet en stor input data størrelse, som kan være grunnen til at implementasjonen ikke fungerte. Ved mer testing av en slik metode vil man kunne få bedre resultater. Kontrastive self-supervised vison transformere (Hua-Bao Ling, 2022) er en mulighet, hvis man vil gå enda dypere.

5.4 Eget arbeid og læring

Læringen og arbeidet var redusert når jeg ikke hadde tilgang til skolens maskiner. Før jeg fikk tilgang var læringen og gjennomføring basert på antagelser og lavnivå tester. Her ble blant annet valg av datasett og grunnleggende metode gjort. Den største andelen av oppgaven ble gjort når jeg fikk tilgang. Dette var blant annet maskinlærings delen og testing. Ved videre arbeid med dyp lærings modeller, er det essensielt med maskin tilgang. Arbeidsmengden ble skjevt fordelt mot slutten av prosjektet, der dette var oppfylt. Fordeling av arbeidstid var et problem jeg hadde gjennom prosessen. Jeg måtte prioritere et ferdig resultat, selv om problemstillingens omfang ble redusert.

Strukturen av arbeidsdager senere i prosessen virket positivt på læring og gjennomføring av prosjektet. Ved å starte dagen med testing og validering av modellen som var trent natten før, gjorde det mulig å se svakheter eller styrker. Hvis problemene var mindre kompliserte leste jeg meg opp, og optimaliserte eller reviderte modell arkitekturen til en forbedret del. Deretter satt jeg på en ny kjøring over natten. Ved større problemer ble dette tatt opp med veileder. Eksempler på slike problemer var, revidering av problemstilling, endring av data størrelse inn i modellen og hvilken øyebevegelse som skulle klassifiseres. I tillegg var gjennomføring på mindre datasett, for eksempel MNIST i iterasjon 1. av prosessen, til hjelp ved valg av fremtidige modeller. Underveis presentasjon av prosjektet ga bedre innblikk i prosessen ved bruk av lavt nivå modeller, spesielt da den ble gjennomført da jeg ikke hadde tilgang til NTNU maskin.

6. Konklusjon

Bruk av en vanlig auto enkoder, dimensjons reduksjon og k-means er ikke en god løsning for problemstillingen. Etter bruk av klynge algoritme, stemte ikke resultatene fra min modell med IVVT algoritmen. Problemstillingen sier at oppgaven handler om klassifisering av øyebevegelser, men ettersom resultatene ikke stemte overens, ble ikke problemstillingen besvart. Dette betyr ikke at lignende ende-til-ende unsupervised modeller ikke er mulige løsninger for problemstillingen. Den originale problemstillingen påpeker at modellen skal testes og kryss valideres mellom flere datasett og tradisjonelle algoritmer. Kryssvalidering ble ikke gjort, utenom IVVT algoritmen, ettersom modellen ikke svarte til problemstilling.

Jeg anbefaler ikke å bruke metoden jeg implementerte ved videre arbeid på denne eller lignende problemstillinger. Bruk av mer avanserte auto enkodere kan være en løsning, eller andre supervised lærings baserte modeller. Dyp lærings baserte modeller påkrever høy datakraft, og dette burde være tilgjengelig gjennom hele prosessen. Hvis ny problemstilling går ut på mer komplekse modeller, anbefaler jeg å ta kontakt med relevante eksperter. Til slutt anbefales en rettet problemstilling, der dette vill gjøre det lettere å estimere tidsbruk.

Bibliografi

Ahmed, M., Seraj, R., & Islam, S. M. S. (2020). The k-means Algorithm: A Comprehensive Survey and Performance Evaluation. *Electronics*, *9*(8). https://doi.org/10.3390/electronics9081295

Alexander, R. G., Macknik, S. L., & Martinez-Conde, S. (2018). Microsaccade Characteristics in Neurological and Ophthalmic Disease. *Frontiers in Neurology*, *9*. https://doi.org/10.3389/fneur.2018.00144

Andrey Vakunov, D. L. (2020). MediaPipe Iris: Real-time Iris Tracking & Depth Estimation [Accessed: 2024-05-06]. https://blog.research.google/2020/08/mediapipe-iris-real-time-iris-tracking.html

Artsiom Ablavatski, I. G. (2019). Real-Time AR Self-Expression with Machine Learning [Accessed: 2024-05-06]. https://blog.research.google/2019/03/real-time-ar-self-expression-with.html

Birawo Birtukan, K. P. (2022). Review and Evaluation of Eye Movement Event Detection Algorithms. *Sensors*, *22*(22). https://doi.org/10.3390/s22228810

Bittencourt J., T. S. e. a., Velasques B. (2013). Saccadic eye movement applications for psychiatric disorders. *Neuropsychiatric disease and treatment*, *9*, 1393–1409. https://doi.org/10.2147/NDT.S45931

Broussard, D., Titley, H., & Heskin-Sweezie, R. (2010). Motor Learning in the Vestibulo-Ocular Reflex. I G. F. Koob, M. L. Moal & R. F. Thompson (Red.), *Encyclopedia of Behavioral Neuroscience* (s. 273–279). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-08-045396-5.00246-3

Collins, T., Semroud, A., Orriols, E., & Doré-Mazars, K. (2008). Saccade Dynamics before, during, and after Saccadic Adaptation in Humans. *Investigative Ophthalmology Visual Science*, *49*(2), 604–612. https://doi.org/10.1167/iovs.07-0753

Croitoru, F.-A., Hondru, V., Ionescu, R. T., & Shah, M. (2023). Diffusion Models in Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(9), 10850–10869. https://doi.org/10.1109/TPAMI.2023.3261988

de-la-Bandera I., M. J. e. a., Palacios D. (2020). Feature Extraction for Dimensionality Reduction in Cellular Networks Performance Analysis. *Sensors*, *20*(23), 6944. https://doi.org/10.3390/s20236944

Du Tran, L. T. e. a., Heng Wang. (2018). A Closer Look at Spatiotemporal Convolutions for Action Recognition.

EL, R. (1985). Normal saccadic velocities. *J Pediatr Ophthalmol Strabismus*. https://doi.org/10.3928/0191-3913-19850101-07

Garbin, S. J., Shen, Y., Schuetz, I., Cavin, R., Hughes, G., & Talathi, S. S. (2019). OpenEDS: Open Eye Dataset.

GoogleCloud. (u.å-a). Machine Learning Clustering Overview [Accessed: 2024-05-06].

GoogleCloud. (u.å-b). What is Unsupervised Learning? [Accessed: 2024-05-06].

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition.

Hua-Bao Ling, D. H. e. a., Bowen Zhu. (2022). Vision Transformer for Contrastive Clustering.

Huey, E. B. (1908). *The Psychology and Pedagogy of Reading*. New York: Macmillan.

Irina Higgins, X. G. e. a., Loic Matthey. (2016). Early Visual Concept Learning with Unsupervised Deep Learning.

Jean Lahoud, F. S. K. e. a., Jiale Cao. (2022). 3D Vision with Transformers: A Survey.

Kapoula, Z., Robinson, D., & Hain, T. (1987). DYNAMIC OVERSHOOT AND POST-SACCADIC DRIFT. I J. O'REGAN & A. LEVY-SCHOEN (Red.), *Eye Movements from Physiology to Cognition* (s. 158–159). Elsevier. https://doi.org/10.1016/B978-0-444-70113-8.50024-2

Koken, P. W., & Erkelens, C. J. (1994). Short Delays of Vergence Eye Movements in Man During Pursuit Tasks in Light and Dark Conditions. I G. d'Ydewalle & J. Van Rensbergen (Red.), *Visual and Oculomotor Functions* (s. 157–168, Bd. 5). North-Holland. https://doi.org/https://doi.org/10.1016/B978-0-444-81808-9.50020-2

Kondratyuk, D., Yuan, L., Li, Y., Zhang, L., Tan, M., Brown, M., & Gong, B. (2021). MoViNets: Mobile Video Networks for Efficient Video Recognition.

Kothari, R., Yang, Z., Kanan, C., Bailey, R., Pelz, J. B., & Diaz, G. J. (2020). Gaze-in-wild: A dataset for studying eye and head coordination in everyday activities. *Sci Rep*, *10*(1), 2539. https://doi.org/10.1038/s41598-020-59251-5

Krauzlis, R. J., & Lisberger, S. G. (1994). Temporal properties of visual motion signals for the initiation of smooth pursuit eye movements in monkeys [PMID: 7965001]. *Journal of Neurophysiology*, *72*(1), 150–162. https://doi.org/10.1152/jn.1994.72.1.150

Kurita, T. (2014). Principal Component Analysis (PCA). I K. Ikeuchi (Red.), *Computer Vision: A Reference Guide* (s. 636–639). Springer US. https://doi.org/10.1007/978-0-387-31439-6_649

Li., D. (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, *29*(6), 141–142. https://doi.org/10.1109/MSP.2012.2211477

Martinez-Conde S, H. D., Macknik S. (2004). The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*. https://doi.org/10.1038/nrn1348

Mestre C, P. J., Gautier J. (2018). Robust eye tracking based on multiple corneal reflections for clinical applications. *J Biomed Opt*. https://doi.org/10.1117/1.JBO.23.3.035001

Mwanga, E. P., Siria, D. J., Mitton, J., Mshani, I. H., González-Jiménez, M., Selvaraj, P., Wynne, K., Baldini, F., Okumu, F. O., & Babayan, S. A. (2023). Using transfer learning and dimensionality reduction techniques to improve generalisability of machine-learning predictions of mosquito ages from mid-infrared spectra. *BMC Bioinformatics*, *24*(1), 11. https://doi.org/10.1186/s12859-022-05128-5

Negi, S., & Mitra, R. (2020). Fixation duration and the learning process: an eye tracking study with subtitled videos. *Journal of Eye Movement Research*, *13*(6). https://doi.org/10.16910/jemr.13.6.1

Neha Sharma, A. M., Vibhor Jain. (2018). An Analysis Of Convolutional Neural Networks For Image Classification [International Conference on Computational Intelligence and Data Science]. *Procedia Computer Science*, *132*, 377–384. https://doi.org/https://doi.org/10.1016/j.procs.2018.05.198

Purves D, F. D., Augustine GJ. (2001a). Neuroscience. 2nd edition. Types of Eye Movements and Their Functions [Accessed: 2023-5-6]. https://www.ncbi.nlm.nih.gov/books/NBK10991/

Purves D, F. D., Augustine GJ. (2001b). Neuroscience. 2nd edition. What Eye Movements Accomplish [Accessed: 2023-5-6]. https://www.ncbi.nlm.nih.gov/books/NBK11156/

Rehman I, M. M., Mahabadi N. (2023). Anatomy, Head and Neck, Eye Fovea [Accessed: 2023-5-6]. https://www.ncbi.nlm.nih.gov/books/NBK482301/

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. I N. Navab, J. Hornegger, W. M. Wells & A. F. Frangi (Red.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (s. 234–241). Springer International Publishing.

Sakshi Indolia, S. M. e. a., Anil Kumar Goswami. (2018). Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach [International Conference on Computational Intelligence and

Data Science]. *Procedia Computer Science*, *132*, 679–688. https://doi.org/https://doi.org/10.1016/j.procs.2018.05.069

Salih Hasan Basna Mohammed, A. A. M. (2021). A Review of Principal Component Analysis Algorithm for Dimensionality Reduction. *2*, 20–30. https://publisher.uthm.edu.my/ojs/index.php/jscdm/article/view/8032

Salvucci Dario D., G. J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications*, 71–78. https://doi.org/10.1145/355017.355028

Sarwinda, D., Paradisa, R. H., Bustamam, A., & Anggia, P. (2021). Deep Learning in Image Classification using Residual Network (ResNet) Variants for Detection of Colorectal Cancer [5th International Conference on Computer Science and Computational Intelligence 2020]. *Procedia Computer Science*, *179*, 423–431. https://doi.org/https://doi.org/10.1016/j.procs.2021.01.025

Steinley, D. (2006). K-means clustering: A half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, *59*(1), 1–34. https://doi.org/10.1348/000711005X48266

Strasburger, H. (2020). Seven Myths on Crowding and Peripheral Vision [PMID: 32489576]. *i-Perception*, *11*(3), 2041669520913052. https://doi.org/10.1177/2041669520913052

Tang, G., Müller, M., Rios, A., & Sennrich, R. (2018). Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures.

Ting Chen, M. N. e. a., Simon Kornblith. (2020). A Simple Framework for Contrastive Learning of Visual Representations.

Tobii. (2024). How do Tobii eye trackers work? [Accessed: 2024-05-06]. https://connect.tobii.com/s/article/How-do-Tobii-eye-trackers-work

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio.

Vesal Sulaiman, M. A., Ravikumar Nishant. (2019). Dilated Convolutions in Neural Networks for Left Atrial Segmentation in 3D Gadolinium Enhanced-MRI. I *Lecture Notes in Computer Science* (s. 319–328). Springer International Publishing. https://doi.org/10.1007/978-3-030-12029-0_35

Wade, N. J. (2010). Pioneers of eye movement research. *Iperception*. https://doi.org/10.1068/i0389

Wong, A. M. F. (2008 mai). 7273The Smooth Pursuit System. I *Eye Movement Disorders*. Oxford University Press. https://doi.org/10.1093/oso/9780195324266.003.0011

Yamashita R., D. R. K. G. e. a., Nishio M. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9

Yangdong He, J. Z. (2019). Temporal Convolutional Networks for Anomaly Detection in Time Series. *Journal of Physics: Conference Series*, *1213*(4), 042050. https://doi.org/10.1088/1742-6596/1213/4/042050

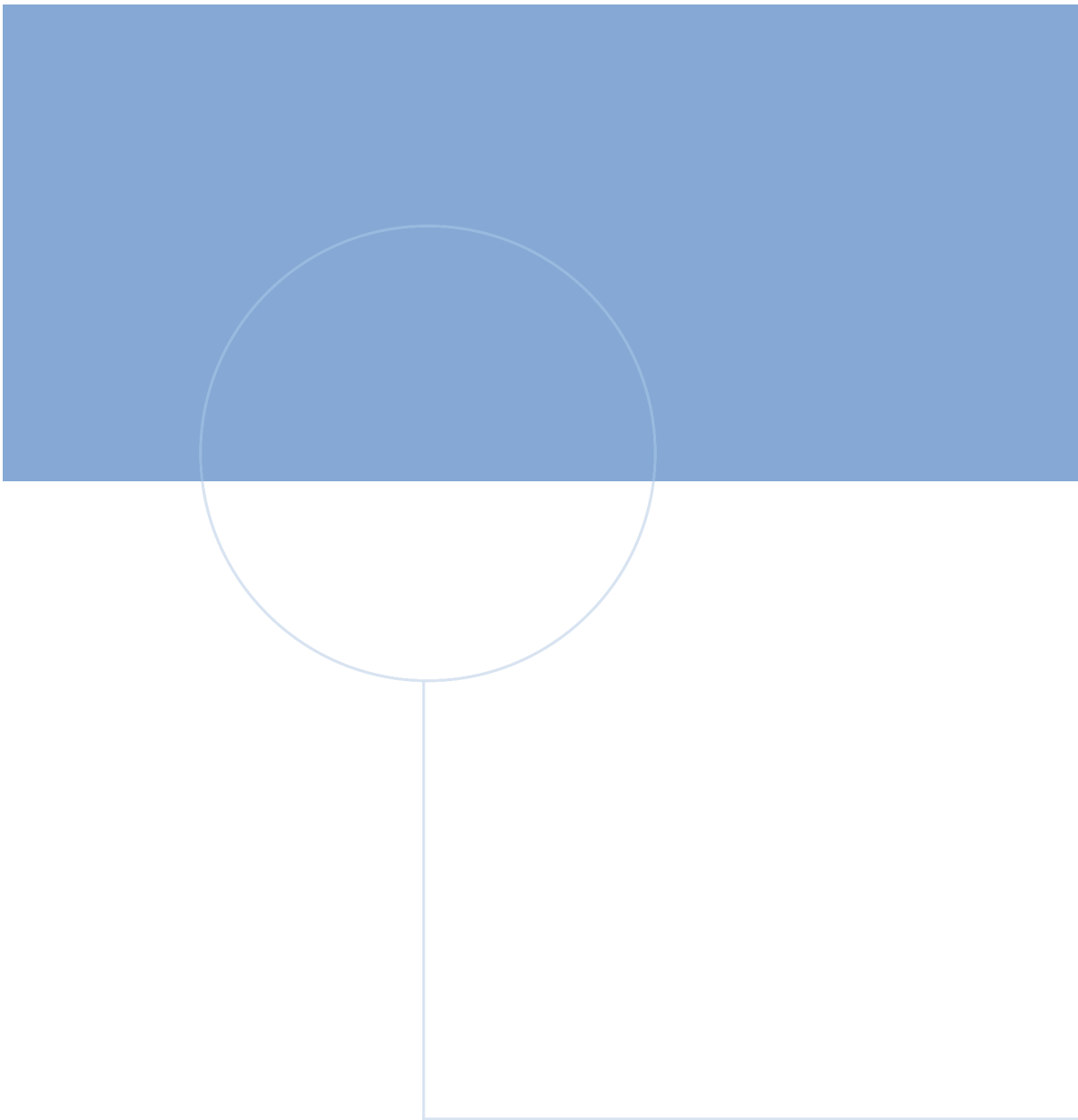
Zemblys R., K. O. e. a., Niehorster D.C. (2018). Using machine learning to detect events in eye-tracking data. *Behavior Research Methods*. https://doi.org/10.3758/s13428-017-0860-3

Zhai, J., Zhang, S., Chen, J., & He, Q. (2018). Autoencoder and Its Various Variants. *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 415–419. https://doi.org/10.1109/SMC.2018.00080

Zhao L, Z. G., Wang Z. (2018). Eye state recognition based on deep. *Multimed Tools Appl*. https://doi.org/10.1007/s11042-017-5380-8

7. Vedlegg

- Prosjekt håndbok
- Møteinnkallinger
- Møtereferater
- Gannt-diagram
- Time liste før påske
- Time liste etter påske
- Orginal Problemstilling
- Presentasjon
- Førprosjekt plan
- Poster presentasjon
- Tsne dimensjon reduksjon plot
- Isomap dimensjon reduksjon plot
- Model arkitekturer (auto enkoder 3,4,5)
- kilde kode
- Repo Readme fil (repo link: https://gitlab.stud.idi.ntnu.no/birkon/eye-movement-classification)



NTNU

Kunnskap for en bedre verden