Systemutviklingsprosjekt - Tolkning og behandling av trafikkskilt på mobiltelefon

Skrevet av Thomas Bjerke og Trym Grande

Forord

Denne rapporten dekker gjennomføringen av systemutviklingsprosjektet i emnet TDAT3022. Under oppstarten av prosjektet fikk vi tildelt flere oppgaveforslag som kunne velges mellom, og valget vårt havnet på oppgaven «tolkning av trafikkskilt på android med OpenCV». Hovedgrunnen til valget var at oppgaven innebar bruk av maskinlæring, og parallelt med TDAT3022 har vi også hatt TDAT3025, anvendt maskinlæring med prosjekt. Vi så derfor denne oppgaven som en mulighet for å bli bedre i to emner på samme tid. Vi fikk også vite at systemutviklingsprosjektet kunne være symbiotisk med prosjektet som skulle gjennomføres i TDAT3025, noe som betydde at vi kunne legge mer tid i ett stort sluttprodukt, fremfor to mindre. Maskinlæringsdelen av oppgaven har altså blitt jobbet med i både dette prosjektet, og maskinlæringsprosjektet.

Gjennomføringen av prosjektet har vært en svært lærerik prosess med tanke på systemutvikling og det å jobbe i team, men også med tanke på maskinlæring.

Vi vil takke veilederen vår, Alexander Holt, som både har vært med på veiledningsmøter, og vært tilgjengelig for å svare på diverse spørsmål gjennom hele gjennomføringen av prosjektet. Han sitter på masse kunnskap om systemutvikling og IT generelt som har kommet godt med. I tillegg vil vi takke Ole Christian Eidheim, som var veilederen vår gjennom maskinlæringsprosjektet og stilte med masse kunnskap som ble brukt i maskinlæringsdelen av oppgaven.

Oppgavetekst

"*Oppgaven går ut på å lage en applikasjon på Android (Java eller Kotlin), hvor man har en telefon oppe på dashboardet i en bil og bruker kameraet til å kontinuerlig filme veien og tolke trafikkskiltene (eller fotobokser) ettersom de dukker opp. Samtidig skal koordinatene til skiltene/fotoboksene bli lagret sammen med skiltet/boksen vha. telefonens GPS. Tolkningen av de*

forskjellige skiltene skal skje ved bruk av OpenCV og trene opp et nevralt nett; vi konsentrerer oss i første omgang kun om fartsgrenser og vil utvide etter hvert.“

Innholdsfortegnelse

Kapittel 1: Introduksjon og relevans

Bakgrunn

Bruksområdene for en applikasjon som tolker trafikkskilt i sanntid er nært sagt uendelige. 3D Motion Technologies, som er oppdragsgivere, oppgir at det er håp om at appen kan brukes til å føre opp skilter i OpenStreetMap. OpenStreetMap er et crowdsourcet alternativ til karttjenester som Google Maps. Dersom brukerne da bare hadde kunnet sette mobilen i dashboardet når de skal på kjøretur, og på den måten registrere alle skiltene på strekningen de kjører, vil dette være mye mer effektivt enn at brukerne manuelt må legge inn hvert skilt de ser.

Det at man enkelt skal kunne lagre skilttype og koordinater til alle skiltene man kjører forbi, gjør appen brukbar for hvem som helst som har interesse av å kartlegge trafikkskilt. Man kunne også for eksempel lagt inn lydmeldinger, slik at appen kan assistere kjøreren av bilen i å følge med på fartsgrensene og hvilke regler som gjelder på veien man kjører på.

Oppbygning

I denne rapporten begynner vi med å se på hvilke løsninger og metoder vi har brukt under gjennomføringen av prosjektet. Vi vil også beskrive hvorfor vi tok de valgene vi tok i et eget kapittel. Deretter, i resultatkapittelet, vil vi legge frem resultatet av prosjektet, før vi går videre til å diskutere resultatet og hvordan det kunne blitt bedre i diskusjonskapittelet. Til slutt blir det hele oppsummert, og vi konkluderer med hvorvidt vi løste problemstillingen, og i korte trekk hvordan.

Problemstilling

Det er flere interessante problemstillinger i denne oppgaven, og den vi har valgt som hovedfokus er følgende:

«*Hvordan kan man bygge og implementere en maskinlæringsmodell til å være rask nok til å tolke trafikkskilt i sanntid, samtidig som den skal kunne kjøre på en mobiltelefons operativsystem og maskinvare?*

Kapittel 2: Teori

Utviklingsprosess

Utviklingsprosessen var iterativ. Det vil si at prosjektløpet ble delt opp i sprinter, hvor teamet i starten av hver sprint bestemte hva som skulle produseres, og på slutten gikk gjennom og testet det som ble produsert. Alle sprintene etter den første bygger på tidligere sprinter. Altså ble det bestemt et minste brukbart produkt, utviklet og testet det, før vi fortsatte å bygge videre på det i neste sprint.

Prinsippene fra smidig og lean programvareutvikling ble også forsøkt fulgt så godt som mulig. Både lean og smidig utvikling har hovedfokus på tilpasningsdyktighet, kundesamarbeid og teamsamarbeid, så de handler om mye av det samme. Lean legger imidlertid aller størst fokus på å unngå å bruke tid på ting som ikke har noen verdi for kunden, noe som det ikke blir sagt noe om i Agile Manifesto [Tesdal 2019]. Nedenfor ligger en oversikt over de prinsippene som har vært viktigst gjennom utviklingsprosessen.

Eliminate waste kommer fra lean utvikling, og handler om å unngå å lage ting som ikke har verdi.

Build quality in kommer fra lean utvikling, og handler om å skrive koden på en sånn måte at det er lett å unngå defekter. I dette prosjektet manifesterte prinsippet seg blant annet i form av at det ble skrevet så lite kode som mulig.

See the whole kommer fra lean utvikling, og handler om å fokusere på systemet som helhet, og det å forstå hele problemdomenet. Avhengigheter til andre systemer er også viktig å sette seg inn i [Tesdal 2019].

Face to face communication kommer fra smidig utvikling og handler om at kommunikasjon ansikt til ansikt er den mest effektive måten å kommunisere på.

Build projects around motivated individuals kommer fra smidig utvikling, og handler om at utviklerne burde få tilgang på ressursene og miljøet de trenger for å løse det de skal løse, og selv finne ut hvordan de vil løse det. Altså lite detaljstyring.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly kommer fra smidig utvikling, og handler altså om å av og til reflektere over hvordan teamet kan bli med effektivt [Beck 2001].

Versjonskontroll

Det ble brukt versjonskontroll under utviklingen. Et versjonskontrollsystem er et system som holder styr på endringer gjort på en eller flere filer over tid, sånn at man kan hente tilbake spesifikke tidligere versjoner [Ernst 2018]. Den viktigste funksjonen som kommer av dette, er at flere utviklere kan jobbe på samme fil samtidig. Det støtter altså ikke-lineær programmering. Dersom flere utviklere jobber på samme kode samtidig, vil det naturlig nok ofte oppstå konflikter når de forskjellige versjonene skal flettes, eller "merges" sammen igjen. Dette kan føre til en "merge konflikt", og må ofte løses manuelt av utviklerne.

Maskinlæring

Dette underkapittelet er gjenbruk fra maskinlæringsprosjektet i TDAT3025.

Metoden som ble brukt for tolkningen av trafikkskilt består at to faser; deteksjon og klassifisering. I deteksjonsfasen prøver vi å oppdage skilt i bildet, men tar ikke hensyn til hvilket skilt det er. Det er det vi gjør i klassifiseringsfasen.

Deteksjon

I deteksjonsfasen bruker vi OpenCV i fem trinn for å oppdage regioner i hver videoframe som ligner på trafikkskilt. OpenCV står for Open Source Computer Vision Library og er, som navnet tilsier, et open source-bibliotek for datasyn og maskinlæring. OpenCV har grensesnitt til mange forskjellige programmeringsspråk, og vi valgte å bruke Python. Vi skal nå se nærmere på hvert av de fem trinnene i deteksjonsfasen.

I første trinn tar vi et bilde fra inputvideoen og øker kontrasten. Dette er både for å enklere kunne skille mellom komponenter med ulik farge, og for å klargjøre for trinn to.

Andre trinn består av å fjerne farger som vi ikke trenger å ta hensyn til videre. Vi fjerner for eksempel svart og grønn fordi ingen norske trafikkskilt har konturer med disse fargene, så vi trenger derfor ikke å sjekke om disse regionene er trafikkskilt, da vi allerede vet at det ikke kan være det.

I tredje trinn bruker vi Laplacian of Gaussian for å finne kanter i bildet, altså for å skille ulike objekter i bildet fra hverandre. Laplacian of Gaussian er en algoritme som detekterer kanter basert på den andrederiverte av et bilde. Altså hvis vi har en graf over intensitetsverdiene til hver piksel i et gråtoneskalabilde, vil LoG se på hvor den andrederiverte av grafen krysser null, og tolke dette området som en kant.

Deretter, i fjerde trinn, gjør vi bildet binært for å enklest mulig kunne finne konturer. Det vil si at vi bruker kantene som vi fant med Laplacian of Gaussian, og gjør kantene hvite, mens resten blir svart. Det gjør at konturer kommer veldig tydelig fram.

Femte og siste trinn består av å gå gjennom konturene fra trinn 4, og finne ut hvilke av disse som ligner på sirkler eller ellipser, nettopp fordi norske fartsgrenseskilt alltid har sirkel-form. Grunnen til at vi også leter etter ellipser er fordi sirkler blir til ellipser dersom de er sett på skrå. Når en sirkel eller ellipse-lignende kontur er funnet, konkluderer vi med at regionen består av et skilt, og sender da den samme regionen fra originalbildet videre til klassifisering.

Klassifisering

Det er i klassifiseringsfasen at skiltene som blir funnet i deteksjonsfasen faktisk blir tolket. Input til klassifiseringen er altså et bilde av noe som deteksjonsfasen tror er et skilt. For å tolke skiltene bruker vi en opptrent SVM-modell. Modellen er trent på et datasett bestående av tyske fartsgrenseskilt som er veldig like, men ikke identiske med norske fartsgrenseskilt.

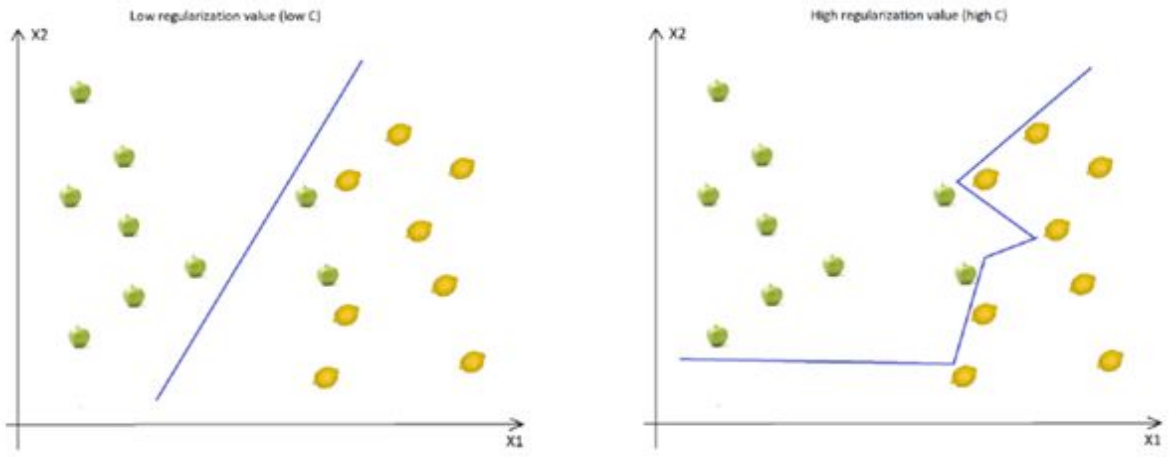
SVM står for Support Vector Machine, og er en supervised maskinlæringsmodell for klassifisering av kategorier. Modellen fungerer slik at den representerer dataen i et vektorrom, og prøver å finne det hyperplanet som på best måte skiller dataen inn i ulike klasser. Det er vanlig blant maskinlæringsalgoritmer å prøve å lære seg de mest vanlige karakteristikkene for hver klasse, for så å basere klassifiseringen på de karakteristikkene. SVM gjør på en måte motsatt.

Den finner de eksemplene som er mest like hverandre fra ulike klasser, og forsøker å finne det optimale hyperplanet for å skille disse. De eksemplene som ligner mest på andre klasser blir kalt support vektorer, og ligger i et n-dimensjonalt vektorrom, hvor n er antall attributter for hvert eksempel. Det optimale hyperplanet er det som gir størst avstand til nærmeste support vektor fra hver klasse. SVM finner to hyperplan som ideelt sett ikke har noen supportvektorer mellom dem, og snittet av disse to blir optimalt hyperplan, ofte kalt decision boundary.

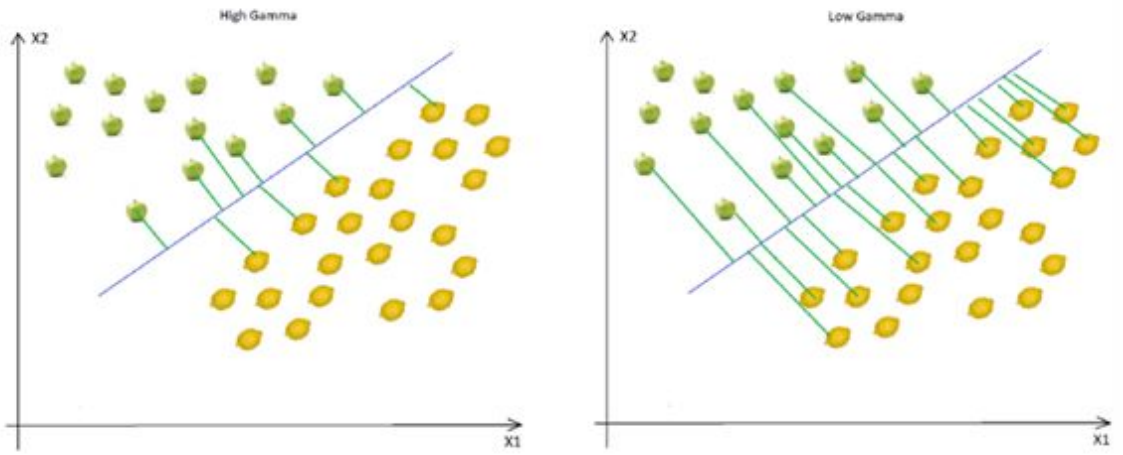
Det er ikke alltid like lett å finne et tydelig skille mellom slike support vektorer, og dette løser SVM ved bruk av kjernefunksjoner. Kjernefunksjoner er mappingsfunksjoner som har som mål å skille de ulike klassene mest mulig fra hverandre. Noen av de vanligste kjernefunksjonene er; polynomisk kjernefunksjon, Gaussisk kjernefunksjon, Radial Basis Function (RBF), Sigmoid, og lineær kjernefunksjon.

Selv med kjernefunksjoner vil det ofte være vanskelig å finne to hyperplan uten noen datapunkter mellom dem, så vi trenger derfor noen trade-offs, toleranse for unntak. Dette styres i SVM av to parametere; regulariseringsparameteren, og gamma.

Reguleringsparameteren kalles i Python for C, og forteller modellen hvor viktig det er å unngå misklassifiseringer. Desto høyere C man bruker, desto mindre blir distansen mellom hyperplanet og supportvektorene. Eksempel med epler og sitroner:



Gammaparameteren bestemmer hvor langt unna det plausible hyperplanet en supportvektor kan ligge og fortsatt bli tatt med i beregningen av hyperplanet. Hvis man bruker høy gammaverdi, vil SVM kun ta med punkter som ligger nærme det plausible hyperplanet i beregningen:



De grønne strekene på bildet brukes for å vise hvilke eksempler som blir tatt med i beregningen.

[Dawson 2019]

Mobilapplikasjon

Hovedverktøyene som har vært brukt under utviklingen og omgjøringen til mobilapplikasjonen, består av "Python for Android", "Buildozer", "Kivy", og "Android Debug Bridge". Alle disse tre henger sterkt sammen, og har derfor stått sentralt under hele utviklingen av prosjektet.

Python for Android

I og med at programmeringsspråket python har blitt brukt i maskinlæringsdelen av prosjektet, med android som operativsystem, var rammeverket Python for Android (også kalt p4a) et passende verktøy. Dette er fordi det fungerer som et abstraksjonslag, eller "lim", mellom python- og android-arkitekturen. Dette gjør at vanlig python-kode kan kjøres på telefon ved å "pakke inn", eller "wrappe" koden på en spesiell måte. Dette systemet har et eget "repo" som blir kontinuerlig oppdatert og vedlikeholdt. Blant annet tillater det for mulighet for at utviklere kan

utvide mengden kompatible moduler ved å inkludere såkalte “recipes”, eller oppskrifter på hvordan bibliotekene skal kompileres ned til en p4a-utgave. Det finnes mange recipes på populære moduler, noe som gjør det teoretisk mulig å kjøre de mest avanserte python-modulene på en telefon. Der det ikke er støtte for en recipe, er det til og med mulig å lage en egen.

[Taylor 2015]

Buildozer

I tillegg til denne innpakningen, trengs det også en slags videre innpakning for å få kildekoden med avhengigheter over til en kjørbar app, eller APK-fil i dette tilfellet, da Android blir brukt for testing. Verktøyet Buildozer er ment for å enkelt kunne lage applikasjoner på den måten, og fungerer for både Android, iOS, Windows, OSX, og Linux. Buildozer støtter Python for Android-innpakningen ved å inkludere “repoet”, som vil si at p4a er innebygd. Dette gjør at man enkelt kan definere hvilke moduler systemet er avhengig av i buildozer.spec-fila. Derfra henter Buildozer tilsvarende recipes fra p4a og installerer dem i p4a-utgaven av systemet.

[PyPi org. 2020]

Kivy

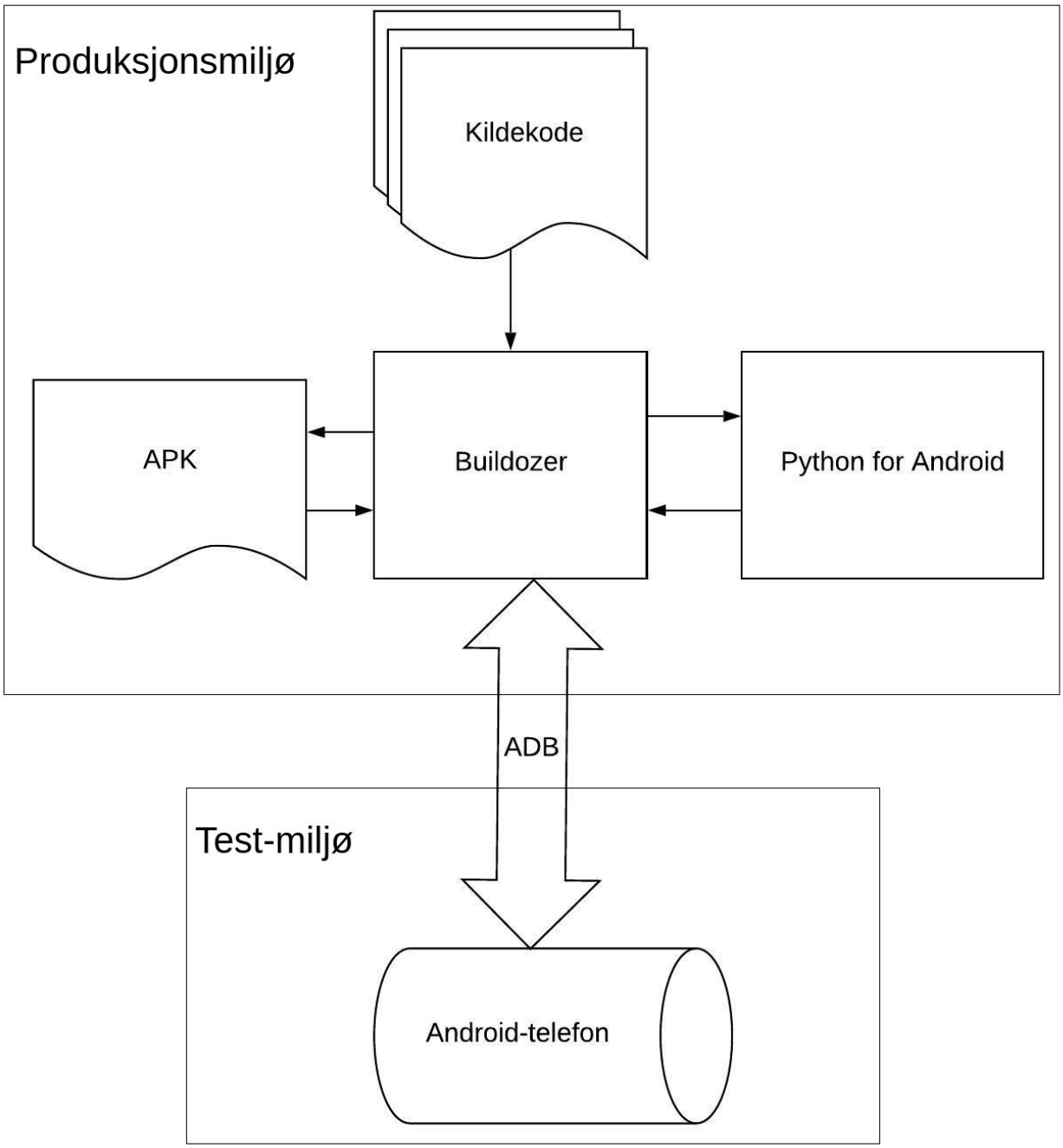
I tillegg, brukes Python-biblioteket Kivy til grafisk brukergrensesnitt som fungerer bra med python når det kommer til applikasjonsutvikling. Dette er utviklet sammen med Python for Android, og er ment til å fungere sammen. [Kivy 2018]

Android Debug Bridge

ADB fungerer ved at klienten, altså produksjonsmaskinen, sender kommandoer til en tjener på Android-systemet (kalt daemon). I dette tilfellet fra Buildozer til Android-telefonen, der kommandoene består av APK-en. I tillegg, brukes brua til debugging, ved at tjeneren på telefonen, returnerer debug logs. Alt dette skjer fysisk over USB. [Google 2020]

Deployment-prosessen

Alle verktøyene beskrevet over henger sammen i produksjonsmiljøet som beskrevet i diagrammet under. Fra toppen, sendes kildekode i form av python og kv (Kivy), sammen med avhengighetene deres inn til Buildozer. Dette behandles i Python for Android der avhengigheter blir installert i form av recipes. Dette blir sendt som en apk fra buildozer til telefonen over ADB via en USB-kabel, som krysser grensen mellom produksjon- og test-miljøet.


Produksjonsmiljø
Kildekode
APK
Buildozer
Python for Android
ADB
Test-miljø
Android-telefon

Kapittel 3: Valg av teknologi og metode

Utviklingsprosess

Teamet bestod av kun to utviklere, så det å finne et bestemt utviklingsrammeverk og følge det etter boken ville vært lite hensiktsmessig, da de fleste utviklingsrammeverk er ment for større team. I stedet ble det plukket ut metoder og prinsipper som ga mening å bruke i små team fra forskjellige rammeverk, som for eksempel SCRUM.

Grunnen til at vi valgte en iterativ utviklingsprosess var først og fremst for å oppnå en jevn, effektiv og ryddig arbeidsflyt.

Arbeids- og rollefordeling

I og med at teamet bestod av kun to personer, ble det ikke bestemt spesifikke roller for hver person. Når noe måtte gjøres, ble det den personen som først hadde ledig tid som gikk inn i rollen det var behov for. Likevel var vi godt kjente med hverandres styrker og svakheter, så det ble også tatt hensyn til under fordeling av arbeid.

Programmeringsspråk

For maskinlæringsmodellen valgte vi å bruke Python som programmeringsspråk. Grunnen til det var hovedsakelig at man i Python har tilgang til en del biblioteker som ville komme godt med i løsningen av prosjektet, som for eksempel NumPy, OpenCV og forskjellige maskinlæringsbiblioteker. I tillegg virket det naturlig å bruke samme programmeringsspråk som vi hadde brukt i undervisningen og øvingene i TDAT3025.

Det offisielle språket for Android er Java, så ettersom vi skrev modellen i Python, måtte vi bruke et grensesnitt for å kunne kjøre den på android-enheter. Dette bydde på store utfordringer. For å få til dette, valgte vi å bruke Kivy, Buildozer og Python for Android. Disse tre fungerer som et bindeledd mellom de to forskjellige arkitekturene, og gjør dette mulig, men på en bekostning av komplisering, som nevnes i diskusjon.

Maskinlæringsmodell

For å tolke skiltene brukes en opptrent SVM-modell. Hovedgrunnen for dette valget var først og fremst hastigheten, kombinert med at det er en simplistisk modell som går relativt raskt å sette opp [Catanzaro 2008]. Dette var viktig da vi hadde begrenset med tid, og at sluttmålet var å få den til å fungere i sanntid.

I oppsettet av SVM-modellen var det, som nevnt i teorikapittelet, tre variabler å ta hensyn til: kjernefunksjon, reguleringsparameteren, og gammaparameteren. I starten ble det bare testet med ulike verdier som virket å gi mening ut ifra teorien om hvordan disse variablene påvirket modellen. Etter hvert fant vi ut at det ville være lurt å lage et gridsearch for å optimalisere.

Gridsearchet vi lagde tok maskinlæringsmodellen og en tabell for hver av variablene som input. Den kryssjekket da variablene mot hverandre, og sjekket hvilken kombinasjon som ga best treffsikkerhet. I starten hadde vi kun med RBF som kjernefunksjon i gridsearchet. Grunnen var at etter å ha lest om de ulike kjernefunksjonene kom vi fram til at det ga mest mening å bruke RBF.

Etterhvert fant vi en vitenskapelig rapport hvor de testet ulike kjernefunksjoner for å se hvilken som ga best nøyaktighet for skiltklassifisering med SVM [Shi 2008]. Der fant de ut at det var lineær kjernefunksjon som ga best nøyaktighet, noe de også ble overrasket over ("Surprisingly, the basic linear kernel performed better than other kernels."). Vi bestemte oss derfor for å likevel ta med andre kjernefunksjoner i gridsearchet, og da den var ferdig å kjøre, viste det seg at det stemte at lineær kjernefunksjon ga best treffsikkerhet. Ved bruk av lineær kjernefunksjon trenger ikke SVM de andre variablene, så de ble derfor ikke tatt hensyn til videre. Lineær kjernefunksjon er også det som gir raskest trening og klassifisering, noe som passet veldig bra for oss i og med at målet var å få modellen til å kjøre på mobiltelefon i sanntid.

Kapittel 4: Resultater

Resultatet av prosjektet er mangelfullt fordi teamet satte seg fast i fjerde sprint grunnet problemer med python for android, kivy og buildozer. Dette førte til at vi ikke fikk hentet bilder fra mobilkameraet. Mer om dette i diskusjonskapittelet.

Vitenskapelige resultater

I og med at det ikke ble funnet noen løsning på problemene med å hente bilder fra mobilkameraet, så ble det heller aldri mulighet for å teste om maskinlæringsmodellen faktisk var effektiv nok til å fungere i sanntid på mobiltelefon. Problemet ligger nå i "cv2.VideoCapture()" i "main.py", der vi prøver å hente inn en bildestrøm fra kameraet. Det virker som om det er et "abstraksjonslag" vi ikke kommer over for å få tilgang på kameaet fordi det har fungert over pc. Applikasjonen får altså trent opp modellen på datasettet, men kræsjer uten Traceback ved nevnt OpenCV-kall.

Det er imidlertid ingen problem å kjøre modellen på datamaskin. Nedenfor er en beskrivelse av resultatene fra kjøring av maskinlæringsmodellen på datamaskin. Dette er gjenbruk fra maskinlæringsprosjektet, og tas med her fordi det ble brukt tid på maskinlæringsdelen i både dette prosjektet og i maskinlæringsprosjektet.

Maskinlæring på datamaskin

For å få til et mål på hvor god modellen var, ble det implementert en metode som viste både treffsikkerhet for datasettet og en confusion matrix. Treffsikkerheten regner ut antall frames der modellen gjetter riktig kontra feil.

Confusion matrixen viser hvordan treffsikkerheten er fordelt over de forskjellige skiltene i datasettet. Den viser label/target for hver rad og prediksjon for hver kolonne. Her kommer altså nøyaktigheten for hvert enkelt skilt fram. Skiltene er sortert slik: [ukjent skilt-lignende objekt, 20, 30, 40, 50, 60, 70, 80, 100, 120].

Trening

Funksjonen trener først opp modellen på datasettet, og tester deretter på en ny, separert del av datasettet. Dette gjøres for å unngå overtilpasning der modellen "husker" bildene i datasettet. Resultatet måles i prosent, og ble i vårt tilfelle 93.56%. Dette reflekterer hvor god modellen er for sitt eget datasett, men ikke nødvendigvis i en reell setting med andre forhold.
Her ser vi stort sett verdier langs diagonalen, som viser god nøyaktighet.

confusion matrix:
[[296 0 0 0 0 0 0 0 0 0]
[0 19 0 0 0 0 0 0 0 0]
[2 0 202 0 8 0 0 2 0 0]
[1 0 0 116 0 0 0 0 0 0]
[0 0 19 0 213 2 0 9 1 0]
[1 0 0 0 2 125 0 10 0 0]
[0 0 0 0 0 0 181 0 0 1]
[1 0 3 0 4 10 0 154 4 1]
[0 0 1 0 2 1 0 4 124 4]
[0 0 0 0 4 0 0 2 7 110]]

Test

Vi gjorde det samme for nøyaktigheten for et tenkt produksjonsmiljø. I dette tilfellet tok vi en video modellen ikke hadde sett før, og lagra alle skiltene i videoen med antatt label. Etter litt manuelt arbeid med å korrigere riktig label, kunne vi da kjøre modellen på nytt med dette test-datasettet, og få nøyaktigheten på samme måte som med funksjonen vi brukte på testdatasettet.

accuracy: 86.31 %

confusion matrix:

```
[[ 1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0]
 [ 0  1  0 65  0  0  0  0  0  0]
 [ 1  1  7  0 109 11  0  5  0  0]
 [ 0  0  1  0  7 63  0  0  0  0]
 [ 1  1  4  0  4  0 69  2  0  0]
 [ 0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  1  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0]]
```

I dette tilfellet ser vi at 50 og 60-skilt blir forvekslet med hverandre omtrent 1/10 av gangene. Grunnen til at 20, 30, 80, 100, og 120 mangler her er fordi vi ikke rakk å samle inn nok testdata i produksjonsmiljø for disse. Disse har allikevel blitt brukt under trening, og mangler kun testing. I og med at dataen allerede har blitt brukt for trening, vil testing med den samme dataen gi en overdreven nøyaktighet grunnet overtilpasning, og er derfor ikke aktuelt.

Se vedlegg for demovideo.

Her er noen bilder fra noen av demovideoene:


40
40


50
50
706
E 6

Ingeniørfaglige resultater

I visjonsdokumentet, som ligger vedlagt, var det oppsatt mål om at sluttproduktet skulle inneholde følgende funksjonelle egenskaper:

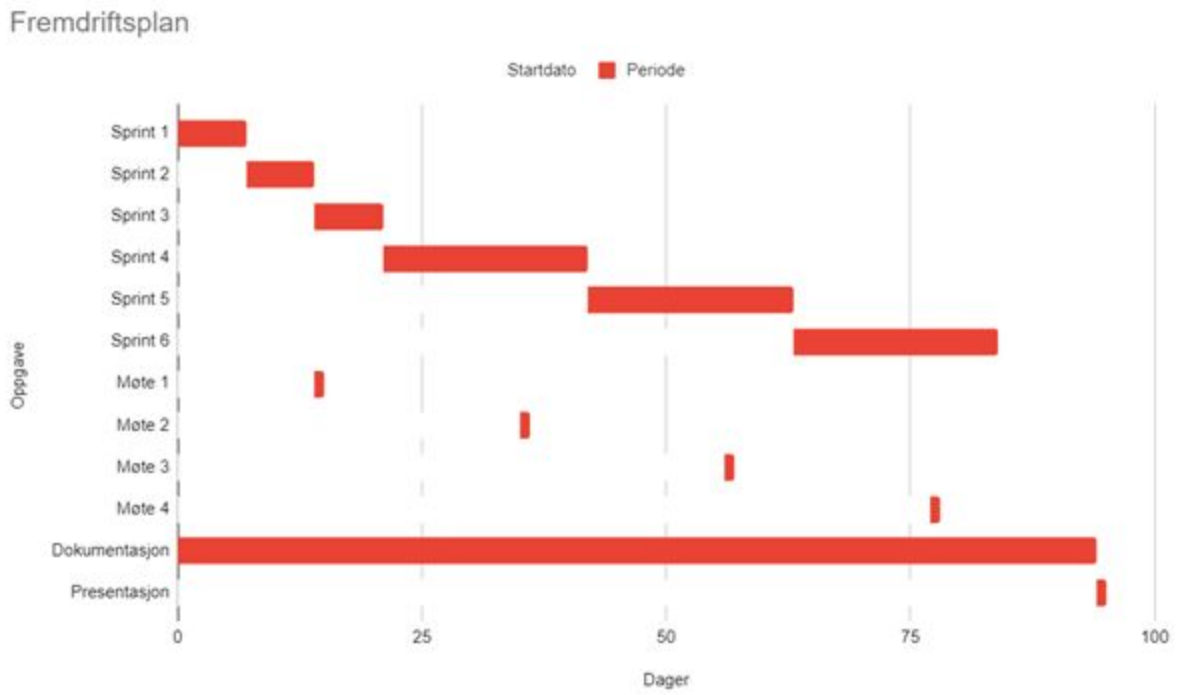
Funksjonelle egenskaper	**Beskrivelse**
Funksjon for filme veien	Appen skal bruke kameraet på telefonen til å filme veien.

Funksjon for å klassifisere trafikkskilt	En maskinlæringsmodell skal ta bilder fra filmingen og klassifisere skiltene til riktig kategori i sanntid
Funksjon for å lagre skilttype og posisjon	Mobiltelefonens GPS skal brukes for å lagre koordinatene til skiltene

Den første funksjonen er altså ikke med i resultatet. Den andre funksjonen, klassifisering av trafikkskilt, fungerer på datamaskin. Det er vanskelig å si noe om hvorvidt klassifiseringen ville fungert godt i sanntid på mobil ettersom vi ikke får testet det uten å løse kamera-problemet. Den siste funksjonen er heller ikke med i resultatet, da planen var å utvikle den etter at kamera-problemet var løst, noe det aldri ble.

Administrative resultater

Framdriftsplanen ligger i prosjekthåndboka som ligger vedlagt, og ser slik ut:



Planen var altså å bruke de tre første sprintene på maskinlæring. Disse sprintene skulle egentlig vare i lenger enn 1 uke hver, men så oppsto muligheten for å "kombinere" maskinlæringsprosjektet med dette prosjektet. Da ble mange av timene som var allokert til maskinlæringsdelen i dette prosjektet flyttet over til maskinlæringsprosjektet, noe som frigjorde mange timer i dette prosjektet som vi heller kunne bruke på app-delen.

De tre første sprintene gikk akkurat slik som tenkt i planen. I sprint 4 derimot, satte vi oss som nevnt fast, og brukte resten av prosjekttiden på å prøve å løse problemet for å komme oss videre. Fremdriftsplanen ble dermed ikke fulgt resten av prosjektet.

Kapittel 5: Diskusjon

Måloppnåelse

Å løse dette problemet har ikke vært like forutsigbart som andre systemutviklingsprosjekt vi har hatt tidligere. Vi har hatt en rekke ulike problemer, som har vært både vanskelige og uforutsigbare å feilsøke. Grunnene til dette vil vi si er noe som kunne vært unngått i et fremtidig prosjekt, og som sannsynligvis hadde ført til et bedre resultat. Dette er derimot enkelt å kommentere i etterkant, når man allerede har vært gjennom det.

Målet i prosjektet ble ikke nådd. Til tross for at timetallene ble fylt opp, har sannsynligvis mye av prosessen ført til lav grad av måloppnåelse. Videre i diskusjonskapitellet ser vi på hva slags problemer prosjektet ble hindret av, og hva som kunne blitt gjort annerledes.

Operativsystem

Til å begynne med, hadde vi regnet med å kunne bruke virtuelle miljø i Windows, noe vi var delvis vandt med fra før gjennom erfaring fra maskinlæringsfaget. Dette viste seg å fungere dårlig, spesielt når det kom til buildozer, som bør kjøres på linux "natively". Dette erfarte vi ved å først bruke en virtuell maskin, som kan føre til uforutsigbare "bugs", samt problemer med kobling mellom Linux-operativsystemet og usb-porter med Android Debug Bridge. Selve installasjonen av operativsystemet viste seg å ikke være like rett fram med windows som parallell-os. Dette har vært grunnet en kombinasjon av lite diskplass på egen pc, samt måten windows-partisjonerer disken, som gjør mye av disken ikke-allokerbar til nytt operativsystem. I vår situasjon måtte man også partisjonere directories, paging-filer o.l. manuelt ved installasjon, som var tidkrevende - spesielt med begrenset diskplass. Senere i prosjektet førte dette til korruptering av hele operativsystemet, der prosjektinstallasjonen ble risikert mistet, men

heldigvis gjenopprettet. Tidsforbruk gikk da også med på diskopprydding i Windows, for å så partisjonere mer plass i Linux.

Avhengigheter

Når vi først var i gang med Buildozer, merket vi med èn gang at dette var litt ustabil programvare, da det fortsatt er i utviklingsfasen "beta", som er hakket før "production/stable". Eksempelvis, fikk vi aldri Traceback for problemet vi fortsatt står fast med angående henting av video. I tillegg til det, er det mye avhengigheter i systemet. Dette inkluderer ting som versjonnummer, moduler, og forskjellige variabler for Android-systemet som OS-versjon, ndk, apk, minimum target osv.

Miljøvariabler

Da vi først satte opp Buildozer, tok det et par forsøk for å få det til å fungere på en gitt maskin. Dette fikk vi til til slutt ved å følge riktig installasjonsveiviser nøye, for å så feilsøke en god stund til installasjonen fungerte. Vedlagt i kildekoden ligger stegene vi gikk gjennom i filen "setup.txt". Når vi framover gjorde endringer i systemet, kunne det plutselig oppstå en eller annen avhengighet e.l. som stoppet hele bygge-prosessen til Buildozer. Dette kunne også skje av enkle ting, som f.eks. opplastning og nedlastning via gitlab, som førte til mye frustrasjon. Etter hvert, fant vi ut at en løsning ofte var å bruke Buildozer sin innebygde "clean"-kommando på produksjonsmiljøet vårt. Dette ville vært mer systematisk og lettere å unngå ved bruk av "Docker" grunnet at systemet er sterkt avhengig av miljøvariabler. Med Docker, har man full kontroll over produksjonsmiljøet ved at installasjonskommandoer kan gjøres systematisk og med gjenopprettingspunkter.

Hva som kunne blitt gjort annerledes

En annen retning for utviklingen, hadde vært å brukt Java som "native" språk, til tross for begrensningene det bringer. Fordelen her er da å slippe et stort abstraksjonslag der man må "porte" om fra en arkitektur til en annen. Dette kan bli en veldig komplisert prosess, noe vi har erfart.

Gruppearbeid

Arbeidet har fungert veldig bra. Vi mener begge at kommunikasjonen har vært god hele veien, grunnet konsekvent oppmøte fra begge parter. Som en gruppe på kun 2 studenter, har det aldri oppstått noe form for misforståelser, krangler, eller konflikter internt.

Kapittel 6: Konklusjon og videre arbeid

Ettersom det aldri ble mulighet for å teste hvordan maskinlæringsmodellen fungerte på mobiltelefon, forblir problemstillingen ubesvart. Vi satte oss fast i fjerde sprint, og brukte resten av prosjekttiden til å prøve å finne en løsning på problemet, noe som sannsynligvis kunne vært unngått ved å ta bedre valg når det kommer til programmeringsspråk i begynnelsen av prosjektet. Eventuelt kunne python fortsatt vært brukt for maskinlæringen, men med et annet grensesnitt til android, da "løsningen" med python for android, kivy, og buildozer bydde på masse problemer.

Maskinlæringsmodellen fungerer imidlertid godt når den kjører på datamaskin, med en treffsikkerhet på over 93% når den testes på samme type bilder som den blir trent på, og over 86% når den testes på bilder fra produksjonsmiljø. I visjonsdokumentet står krav om 70% treffsikkerhet, som altså er oppnådd med god margin ved å bruke en SVM-modell med lineær kjernefunksjon.

Referanser

[Beck 2001] Beck, K., et al. Manifesto for Agile Software Development. Agile Alliance. http://agilemanifesto.org/, sist besøkt 12.12.2020

[Catanzaro 2008] Catanzaro, Bryan & Sundaram, Narayanan & Keutzer, Kurt. (2008). Fast support vector machine training and classification on graphics processors. Proceedings of the 25th International Conference on Machine Learning. 104-111. 10.1145/1390156.1390170.

[Dawson 2019] Carl Dawson: SVM Parameter Tuning. https://towardsdatascience.com/a-guide-to-svm-parameter-tuning-8bfe6b8a452c, sist besøkt 04.11.2020

[Ernst 2018] Michael Ernst: Version control concepts and best practices. https://homes.cs.washington.edu/~mernst/advice/version-control.html, sist besøkt 12.12.2020

[Shi 2008] Shi, M., Wu, H., Fleyeh, H. Support vector machines for traffic signs recognition https://ieeexplore.ieee.org/document/4634347, sist besøkt 06.11.2020

[Tesdal 2019] Nils Tesdal: Systemutviklingsmetodikk https://ntnu.blackboard.com/webapps/blackboard/execute/content/file?cmd=view&content_id=_541618_1&course_id=_12239_1, sist besøkt 10.12.2020

[Kivy 2014] Kivy's Developers: Welcome to Buildozer's documentation!

https://buildozer.readthedocs.io/en/latest/, sist besøkt 15.12.2020

[PyPi org. 2020] PyPi: buildozer 1.2.0

https://pypi.org/project/buildozer/, sist besøkt: 15.12.2020

[Taylor 2015] Alexander Taylor: Recipes

https://python-for-android.readthedocs.io/en/latest/recipes/, sist besøkt: 15.12.2020

[Taylor 2015] Alexander Taylor: python-for-android

https://python-for-android.readthedocs.io/en/latest/, sist besøkt: 15.12.2020

[Google 2020] Google Developers: Android Debug Bridge (adb)

https://developer.android.com/studio/command-line/adb, sist besøkt: 15.12.2020

[Kivy 2018] The Kivy Authors: Welcome to Kivy

https://kivy.org/doc/stable/, sist besøkt: 15.12.2020

[Wikipedia 2020] Wikipedia: Kivy (framework)

https://en.wikipedia.org/wiki/Kivy_(framework), sist besøkt: 15.12.2020

Vedlegg I - Visjonsdokument

Trafikkskilt OpenCV
Visjonsdokument

Versjon 2.2

Skrevet av Thomas Bjerke og Trym Grande

Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
14.09.2020	1.0	Første utkast	Thomas Bjerke og Trym Grande
16.09.2020	2.0	Andre utkast etter møte med veileder	Thomas Bjerke og Trym Grande
25.09.2020	2.1	La til ikke-funksjonelle egenskaper	Thomas Bjerke
10.10.2020	2.2	Små endringer, lagt til interessenter	Trym Grande

Innholdsfortegnelse

1. Innledning

Dette dokumentet beskriver krav til systemutviklingsprosjektet i emnet TDAT3022 for gruppe 12.

2. Sammendrag problem og produkt

2.1 Problemsammendrag

"Oppgaven går ut på å lage en applikasjon på Android (Java eller Kotlin), hvor man har en telefon på oppe på dashboardet i en bil og bruker kameraet til å kontinuerlig filme veien og tolke trafikkskiltene (eller fotobokser) ettersom de dukker opp. Samtidig skal koordinatene til skiltene/fotoboksene bli lagret sammen med skiltet/boksen vha. telefonens GPS. Tolkningen av de forskjellige skiltene skal skje ved bruk av OpenCV og trene opp et nevralt nett; vi konsentrerer oss i første omgang kun om fartsgrenser og vil utvide etter hvert. "

Problem med	Å kartlegge trafikkskilt, deres betydning, og posisjon
berører	alle som har interesse av å kartlegge trafikkskilt. Eksempel: OpenStreetMap
som resultatet av dette	mangler informasjon om fartsgrenser og andre trafikkregler i systemer som OpenStreetMap
en vellykket løsning vil	gjøre det mulig å automatisk kartlegge trafikkskiltene man kjører forbi

2.2 Produktsammendrag

For	*3D Motion Technologies*
som	*har behov for automatisk innfylling kartdetaljer*
Vår app	*er vår automatiske kartleggingstjeneste for skilt*

som	*automatisk gjenkjenner og trafikkskilt og lagrer deres posisjon*
I motsetning til	*å manuelt lagre skiltenes betydning og posisjon*
Har vårt produkt	*en maskinlæringsmodell som tolker skiltenes betydning, samt at posisjonen blir lagret*

3. Overordnet beskrivelse av interessenter og brukere

3.1 Oppsummering interessenter

Navn	**Utdypende beskrivelse**	**Rolle under utviklingen**
Prosjektteam	Utviklere av systemet	Står for selve utviklingen av systemet.
Veileder	Faglærer	Veilede prosjektteamet under gjennomføringen av prosjektet
Oppdragsgiver	Produkteier	Det er oppdragsgiveren som er "kunden". Gir krav til sluttproduktet

3.2 Oppsummmering brukere

Navn	**Utdypende beskrivelse**	**Rolle under utviklingen**	**Representert av**
	Forklar hvilken rolle denne brukeren spiller i dagens system eventuell annen viktig informasjon om denne brukeren	hvilken rolle vil han ha under utviklingen av systeme?	seg selv eller en annen bruker eller interessent?

3.3 Brukermiljøet

Systemet vil kjøre kontinuerlig på en android-telefon som er festet i dashbordet til et kjøretøy i trafikken.

3.4 Sammendrag av brukernes behov

Behov	**Prioritet**	**Påvirker**	**Dagens løsning**	**Foreslått løsning**
Filme veien	høy	kartlegging og tolkning av skilt	ingen	Androidapplikasjon som bruker telefonens kamera til å filme veien
Få fartsgrenseskiltene tolket	høy	kartlegging av skilt	ingen	Lage maskinlæringsmodell som klassifiserer skiltene
Få fartsgrenseskiltenes posisjon lagret	høy	kartlegging av skilt	ingen	Bruke mobilens GPS til å lagre koordinatene
Få andre trafikkskilt tolket	middels	kartlegging av skilt	ingen	Utvide modellen til å kunne klassifisere andre typer skilt

4. Produktoversikt

4.1 Produktets rolle i brukermiljøet

Produktet har som rolle å gjøre følgende:

- Filme langs veien med mobilkameraet for å så prosessere imøtekommende skilt.

4.2 Forutsetninger og avhengigheter

Applikasjonen kan kun kjøres på mobiltelefoner med android operativsystem. Ettersom mobilen skal filme veien fra dashbordet, er det viktig at den er festet godt slik at bildet blir så bra som mulig. I tillegg må mobiltelefonens kamera være godt nok til at modellen klarer å gjenkjenne skiltene.

Det er også viktig at mobilens ytelse er høy nok til å kunne behandle bildene fort nok.

5. Produktets funksjonelle egenskaper

Systemet skal kunne kontinuerlig se etter, oppdage, og gjenkjenne skilt samtidig som brukeren opererer et kjøretøy. Skilt-dataen som samles inn skal videresendes til videre lagring og potensiell behandling.

Funksjonelle egenskaper	**Beskrivelse**
Funksjon for filme veien	Appen skal bruke kameraet på telefonen til å filme veien.
Funksjon for å klassifisere trafikkskilt	En maskinlæringsmodell skal ta bilder fra filmingen og klassifisere skiltene til riktig kategori i sanntid
Funksjon for å lagre skilttype og posisjon	Mobiltelefonens GPS skal brukes for å lagre koordinatene til skiltene

6. Ikke-funksjonelle egenskaper og andre krav

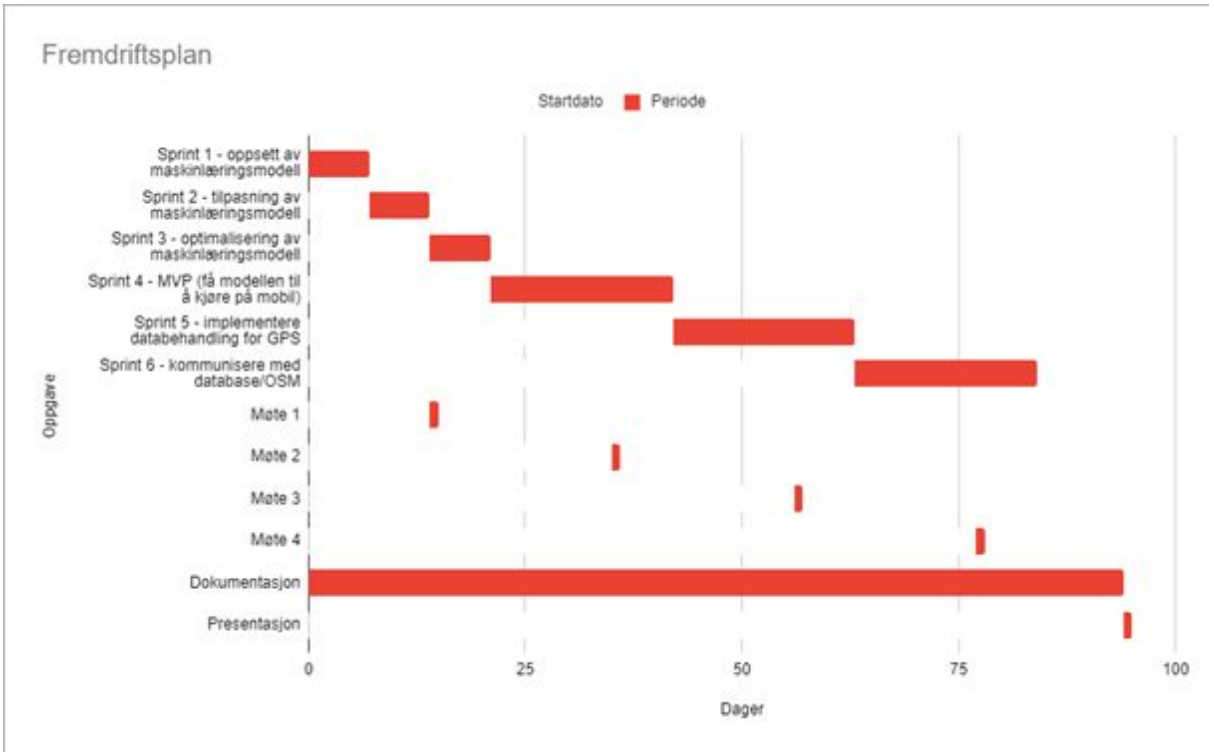
Appen skal være brukervennlig og ha et intuitivt design. Det skal være enkelt for sluttbrukeren å forstå hvordan man bruker appen.

Ikke-funksjonelle egenskaper	**Beskrivelse**
Brukervennlig design	Appens brukergrensesnitt skal være intuitiv, som vil si at det skal være lett for sluttbrukeren å forstå hvordan appen skal brukes
Tilstrekkelig treffsikkerhet	Treffsikkerheten til modellen som klassifiserer skiltene må være høy nok, rundt 70% i produksjonsmiljø

Vedlegg II - Prosjekthåndbok

Framdriftsplan – Gannt-diagram

Oppgave	Startdato	Sluttdato	Periode	
Sprint 1 - oppsett av maskinlæringsmodell	10/9/20	17/9/20	7	dager
Sprint 2 - tilpasning av maskinlæringsmodell	17/9/20	24/9/20	7	dager
Sprint 3 - optimalisering av maskinlæringsmodell	24/9/20	1/10/20	7	dager
Sprint 4 - MVP (få modellen til å kjøre på mobil)	1/10/20	22/10/20	21	dager
Sprint 5 - implementere databehandling for GPS	22/10/20	12/11/20	21	dager
Sprint 6 - kommunisere med database/OSM	12/11/20	3/12/20	21	dager
Møte 1	24/9/20	25/9/20	1	dager
Møte 2	15/10/20	16/10/20	1	dager
Møte 3	5/11/20	6/11/20	1	dager
Møte 4	26/11/20	27/11/20	1	dager
Dokumentasjon	10/9/20	13/12/20	94	dager
Presentasjon	13/12/20	14/12/20	1	dager



Innkalling til møte 1 i Team 12 Trondheim, 16.09.2020

Møteinnkallingen går til:
Alexander Holt, Trym Grande, Thomas Bjerke

Tid og sted: Onsdag 16.09.2020 klokken 14.00 – 14.30. Digitalt møte på zoom.

Agenda:

Saknr	Saker
01	Tid vi skal bruke på prosjektet
02	Trenger vi brukerkontoer?
03	Hvilket språk til maskinlæringsdelen?
04	Skal dataen sendes til OpenStreetMap eller lagres i egen database?
05	Utviklingsrammeverk?
06	Hva slags testing kreves?
07	Visjonsdokument

Det bli ingen pauser eller servering underveis i møtet.

Ta kontakt med Thomas Bjerke (thombje@ntnu.no / tlf 905 92 427) for å gi beskjed dersom du er/ blir forhindret fra å møte.

Velkommen!

Gruppe 12

Møtereferat til møte 1 i Team 12

Tid og sted: Onsdag 16.09.2020 klokken 14.00 – 14.30. Digitalt møte på zoom.
Deltakere tilstede: Alexander Holt, Trym Grande, Thomas Bjerke

1. **Kunngjøringer**

    - Spørsmål om brukerkontoer ble besvart til å foreløpig se bort ifra brukerkontoer fordi det ikke vil være relevant for systemet.
    - Språk for maskinlæringsdelen er satt til valgfritt.
    - Innsamlet data må ikke nødvendigvis implementeres med OpenStreetMap, men kan gjøres senere i prosjektet dersom det lar seg gjøre.
    - Utviklingsrammeverk er valgfritt, men valget må begrunnes i systemdokumentasjonen.
    - Ingen spesiell testing kreves, men kan være greit å ha med dersom det blir tid til det.
    - Mobiltelefon til testing blir teammedlemmenes egen i første omgang, men kan oppgraderes gjennom universitetet dersom det viser seg å kreves for god nok ytelse i applikasjonen.

2. **Diskusjon**

    - Ingen videre diskusjon ble gjort denne gangen, kun avklaringer for oppstart av prosjektet. Alle spørsmål ble besvart.

3. **Status**

    - Status ble heller ikke diskutert da prosjektet fortsatt er i oppstartsfasen.

Møte hevet
Møtet ble hevet 14:30.

Referat sendt av: Trym Grande

Godkjent av: Thomas Bjerke

Innkalling til møte 2 i Team 12 Trondheim, 15.11.20

Møteinnkallingen går til:
Alexander Holt, Trym Grande, Thomas Bjerke

Tid: mandag 16.11.20 kl. 12.00-12.30.
Sted: Digitalt møte over nett (plattform avtalt over e-mail).

Agenda:

Saknr.	Saker
1	Koordinering med maskinlæringsprosjekt
2	Oppdatering på prosjektfremgang
3	Tilbakemelding fra faglærer

Det bli ingen pauser eller servering underveis i møtet.

Ta kontakt med Thomas Bjerke (thombje@ntnu.no / tlf 905 92 427) eller Trym Grande (trymg@ntnu.no / tlf 404 75 830) for å gi beskjed dersom du er/blir forhindret fra å møte.

Velkommen!

Mvh Gruppe 12

Møtereferat til møte 2 i Team 12

Åpner
Møtet til SU3-12 ble startet 12:00, 16.11.2020 på jitsi.

Deltakere tilstede: Alexander Holt, Trym Grande, Thomas Bjerke

Godkjenning av saksliste
Sakslisten ble enstemmig godkjent som distribuert.

Åpne saker
Koordinering med maskinlæringsprosjekt – Trym og Thomas opplyser om at Ole Christian har sendt forslag til hvordan prosjektet kan deles opp. Mailen fra Ole Christian leses opp, og Alexander påpeker at vi må sørge for å fylle opp timene i begge fag.

Oppdatering på prosjektframgang – Trym og Thomas viser en demofilm

Tilbakemelding fra faglærer - Får tilbakemelding om at kamera burde holdes stabilt i dashboardet og ikke beveges mot hvert skilt. Alexander spør hvilken type enhet koden nå kjører på. Trym og Thomas svarer at den foreløpig kjører på PC, men at de jobber med å bruke buildozer og kivy for å få kjørt det på mobil. Alexander liker egentlig ikke Kivy, men nå er det uansett for sent å begynne med en annen løsning. Dersom det hadde blitt noe av det ene møtet som egentlig var avtalt tidligere, kunne dette heller ha kommet fram der, og vi kunne ligget bedre an. Får til slutt beskjed om å fortsette å jobbe med å få kivy til å fungere.

Møte hevet
Møtet ble hevet 12:30.

Referat sendt av: Thomas Bjerke

Godkjent av: Trym Grande

Innkalling til møte 3 i Team 12 Trondheim, 9.12.20

Møteinnkallingen går til:
Alexander Holt, Trym Grande, Thomas Bjerke

Tid: onsdag 9.12.20 kl. 14.00-14.30.
Sted: Digitalt møte over nett (https://meet.idi.ntnu.no/trym).

Agenda:

Saknr.	Saker
1	Oppdatering på prosjektfremgang
2	Dokumentasjon
3	Tilbakemelding fra faglærer

Det bli ingen pauser eller servering underveis i møtet.

Ta kontakt med Thomas Bjerke (thombje@ntnu.no / tlf 905 92 427) eller Trym Grande (trymg@ntnu.no / tlf 404 75 830) for å gi beskjed dersom du er/blir forhindret fra å møte.

Velkommen!

Mvh Gruppe 12

Møtereferat til møte 3 i Team 12

Tid: onsdag 9.12.20 kl. 14.00-14.30.
Sted: Digitalt møte over nett (https://meet.idi.ntnu.no/trym).

Deltakere tilstede: Alexander Holt, Trym Grande, Thomas Bjerke

1. **Kunngjøringer**

    - Presentasjon for prosjektet ble avtalt å gjennomføre til planlagt tid.

2. **Diskusjon**

    - Ingen diskusjon ble gjort da begge partene er enige rundt prosjektet.

3. **Status**
    - Trym og Thomas oppdaterer om lite prosjektfremgang grunnet kompliseringer med sanntidsoppkobling til mobilkamera. Det opplyses også om at et tidlig valg om å bruke "buildozer" sansynligvis har ført til flere kompliseringer enn ved å gjennomføre prosjektet gjennom Androids "native" språk, Java (kontra python).
    - Alexander gir tilbakemelding om at man enten kan fortsette, og prøve videre med andre kamera-løsninger, eller eventuelt "kaste inn håndkleet" dersom det virker håpløst. Uansett sier han at det vil være viktig å få med begrunnelse for valget og diskutere hva man kunne gjort annerledes i hovedrapporten.
    - Videre spør Teamet om det vil være aktuelt å ha med alt av dokumentasjon det er oppgitt i prosjekthåndboka, og Alexander svarer at "en mal er en mal", som vil si at irrelevant stoff kan utelukkes, og ting kan legges til.

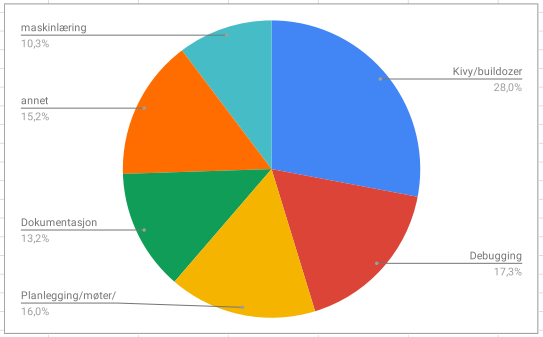
Møte hevet
Møtet ble hevet 12:30.

Referat sendt av: Trym Grande

Godkjent av: Thomas Bjerke

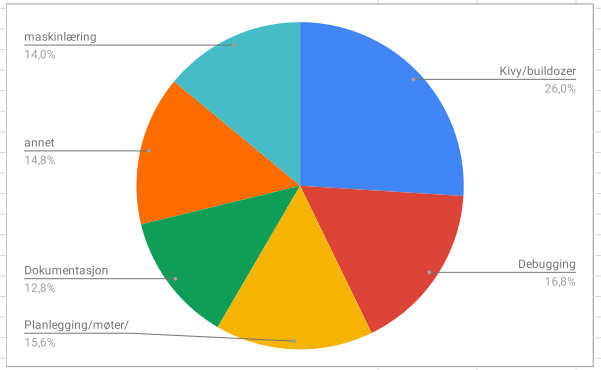
Timeliste for Thomas

Uke	Dato	Kivy/buildozer	Debugging	Planlegging/møter/research	Dokumentasjon	annet (administrasjon/git/miljø)	maskinlæring	Totalt	Kommentar
37	10/9/20			4	2		4	10	research, visjonsdokument, teste maskinlæringsløsninger
37	11/9/20			4	2		4	10	research, wireframes, teste maskinlæringsløsninger
38	16/9/20			7		1	1	9	Leting etter datasett, wireframes, research
38	18/9/20				1	4	4	9	implementere datasett
39	21/9/20		2		1		6	9	skriving av modell/bug fixing
39	23/9/20		2	1	2	1	5	11	skriving av modell/bug fixing
40	28/9/20			4		3		7	sette opp miljø i wsl (fungerte treigt), planlegge merging av ml og app, bygd python3.5 fra source for kjøring av kildekode (viste seg å ikke trenge å være v3.5)
40	30/9/20	3	3	3				9	utforske videre utvikling, installere og teste ut kivy (kivy-app fra play store som launcher, broken, prøvde uoffisiell også)
41	5/10/20		3			3		6	installert kivy på linux vm, fungerte ikkke, må installere på native linux, rydda opp disk for installering
41	6/10/20		3					3	problemer med partisjonering til dual boot pga. windows-filer
42	12/10/20	2	1			5		8	installert linux dual boot og starta med kivy-installasjon
42	14/10/20	2	2	1		3		8	partisjonering for diskplass i linux, installering av buildozer debugging
43	19/10/20	1	4			4	1	10	reinstallert alt, hellIo world app fungerer på tlf med python, buildozer, kivy, git
43	21/10/20					5		5	git
43	23/10/20			1	3			4	dokumentasjon, møteplanlegging
44	26/10/20	9						9	Installeringer og feilsøking av installeringene
44	29/10/20	1						1	Installeringer
45	2/11/20	3		1		3		7	starta på app-integrering
45	4/11/20	1	4	3				8	fortsatt med app-integrering (opencv/androidSDK) (ml -> kivy/buildozer)
46	9/11/20	4		3				7	fortsatt med app-integrering, reinstallerte buildozer pga. byggefeil
46	11/11/20	3		2	2			7	fortsatt med app-integrering, feil med opencv (64 bit istedenfor 32 bit), skjønner ikke recipes, default opencv-recipe har feil versjon kanskje?
47	16/11/20	6		1	1			8	møteinnkalling sys, ukerapport, fortsatt med app-integrering, feil med opencv (64 bit istedenfor 32 bit), testa med forskjellige konfigurasjoner uten lykke (får ikke lenger kjørt hello world)
47	18/11/20	3	2	1		2		8	fiksa buggen (reinstallere .buildozer/endre arkitektur), lagt til datasett i app, møte, git, kræsjer ved video display
48	23/11/20		2			2		4	implementert accuracy-funksjon, git, debug av buildozer (får ikke traceback)
49	28/11/20							0	implementert forvirrings-matrise-funksjon
49	29/11/20		8					8	debugging med åpning av video/kamera i kivy, får fortsatt ikke åpna, men vet at traceback mangler kun i dette tilfellet
49	30/11/20	8						8	debug cv2.videocapture(), fungerer ikke i android, svart skjerm
49	1/12/20	6		2				8	research andre kameraløsninger, camera api med plyer
49	2/12/20	3						3	android manifest, fileprovider
49	3/12/20	3	4			1		8	fileprovider gitt opp, plyer fungerer uten fileprovider på samsung med lesing/skriving til disk
49	4/12/20	2						2	testet lesing og skriving til fil, skjer 1 gang per 2 sek
49	5/12/20	2						2	Prøvd uten lesing/skriving
49	6/12/20		2	1				3	prøvd https://github.com/liyuanrui/kivy-for-android-opencv-demo, bug med image, kan være pga. buildozer-installasjonen
50	7/12/20	2						2	siste forsøk med kivy
50	8/12/20	2			2			4	dokumentasjon
50	9/12/20	2			2			4	dokumentasjon
50	10/12/20				2			2	dokumentasjon
50	11/12/20				2			2	dokumentasjon
50	12/12/20				2			2	dokumentasjon
50	13/12/20				2			2	dokumentasjon
50	14/12/20				6			6	dokumentasjon
50	15/12/20							0	
Totalt:		68	42	39	32	37	25	243	
		Kivy/buildozer	Debugging	Planlegging/møter/research	Dokumentasjon	annet (administrasjon/git)	maskinlæring		



Timeliste for Trym

Uke	Dato	Kivy/buildozer	Debugging	Planlegging/møter/research	Dokumentasjon	annet (administrasjon/g	maskinlæring	Totalt	Kommentar
37	10/9/20			4	2		4	10	research, starte med dokumentasjon, teste ut maskinlæringsløsninger
37	11/9/20			4	2		4	10	research, starte med dokumentasjon, teste ut maskinlæringsløsninger
38	16/9/20			7		1	1	9	finne datasett
38	18/9/20				1	4	4	9	implementere datasett
39	21/9/20		2		1		6	9	skriving av modell/bug fixing
39	23/9/20		2	1	2	1	5	11	skriving av modell/bug fixing
40	28/9/20			4		3		7	sette opp miljø i wsl (fungerte treigt), planlegge merging av ml og app, bygd python3.5 fra source for kjøring av kildekode (viste seg å ikke trenge å være v3.5)
40	30/9/20	3	3	3				9	utforske videre utvikling, installere og teste ut kivy (kivy-app fra play store som launcher, broken, prøvde uoffisiell også)
41	5/10/20		3			3		6	installert kivy på linux vm, fungerte ikkke, må installere på native linux, rydda opp disk for installering
41	6/10/20		3					3	problemer med partisjonering til dual boot pga. windows-filer
42	12/10/20	2	1			5		8	installert linux dual boot og starta med kivy-installasjon
42	14/10/20	2	2	1		3		8	partisjonering for diskplass i linux, installering av buildozer debugging
43	19/10/20	1	4			4		9	reinstallert alt, helllo world app fungerer på tlf med python, buildozer, kivy, git
43	21/10/20					5		5	git
43	23/10/20			1	3			4	dokumentasjon, møteplanlegging
44	26/10/20	8						8	importere/formatere datasett
44	29/10/20	8						8	importere/formatere datasett
45	2/11/20	4		1		3		8	starta på app-integrering
45	4/11/20	1	4	3				8	fortsatt med app-integrering (opencv/androidSDK) (ml -> kivy/buildozer)
46	9/11/20	4		3				7	fortsatt med app-integrering, reinstallerte buildozer pga. byggefeil
46	11/11/20	3		2	2			7	fortsatt med app-integrering, feil med opencv (64 bit istedenfor 32 bit), skjønner ikke recipes, default opencv-recipe har feil versjon kanskje?
47	16/11/20	3		1	1			5	møteinnkalling sys, ukerapport, fortsatt med app-integrering, feil med opencv (64 bit istedenfor 32 bit), testa med forskjellige konfigurasjoner uten lykke (får ikke lenger kjørt hello world)
47	18/11/20	3	2	1		2		8	fiksa buggen (reinstallere .buildozer/endre arkitektur), lagt til datasett i app, møte, git, kræsjer ved video display
48	23/11/20		2			2	5	9	implementert accuracy-funksjon, git, debug av buildozer (får ikke traceback)
49	28/11/20						5	5	implementert forvirrings-matrise-funksjon
49	29/11/20		8					8	debugging med åpning av video/kamera i kivy, får fortsatt ikke åpna, men vet at traceback mangler kun i dette tilfellet
49	30/11/20	4						4	debug cv2.videocapture(), fungerer ikke i android, svart skjerm
49	1/12/20	2		2				4	research andre kameraløsninger, camera api med plyer
49	2/12/20	3						3	android manifest, fileprovider
49	3/12/20	3	4			1	1	9	fileprovider gitt opp, plyer fungerer uten fileprovider på samsung med lesing/skriving til disk
49	4/12/20	3						3	testet lesing og skriving til fil, skjer 1 gang per 2 sek
49	5/12/20	2						2	Prøvd uten lesing/skriving
49	6/12/20		2	1				3	prøvd https://github.com/liyuanrui/kivy-for-android-opencv-demo, bug med image, kan være pga. buildozer-installasjonen
50	7/12/20	2						2	siste forsøk med kivy
50	8/12/20	2			2			4	dokumentasjon
50	9/12/20	2			2			4	dokumentasjon
50	10/12/20				2			2	dokumentasjon
50	11/12/20				2			2	dokumentasjon
50	12/12/20				2			2	dokumentasjon
50	13/12/20				2			2	dokumentasjon
50	14/12/20				6			6	dokumentasjon
50	15/12/20							0	
Totalt:		65	42	39	32	37	35	250	
		Kivy/buildozer	Debugging	Planlegging/møter/research	Dokumentasjon	annet (administrasjon/g	maskinlæring		



Ukerapporter

Ukenummer	Gjennomført	Prosjektstatus	Problemer	Tiltak	Oppgaver for neste uke
37	starte på dokumentasjon, utforske generelle implementasjoner, sette opp prosjekt med opencv og eksempelkode	Starta med dokumentasjon og en mulig løsning	Vet ikke hvor vi skal starte	Utforske mulige løsninger	Fortsette å utforske mulige løsninger
38	lage branches på gitlab, starte på gui/wireframes, planlegging, møte, testing og implementasjon av eksempelkode	Fortsatt med prosjektoppstart	Vet ikke hva slags løsning som fungerer	Fortsette å utforske mulige løsninger	Jobbe videre med prosjektoppstart
39	sette opp (virtuelt) lokalt og remote miljø for kodekjøring	satt opp prosjektmiljø og jobbet videre med mulig løsning	prosjektmiljø fungerte dårlig	bytte til alternative miljø	sette opp nyttt miljø, jobbe videre med løsningen
40	sette opp miljø i wsl, planlegge merging av ml og app, utforske videre utvikling, installere og teste ut kivy (kivy-app fra play store som launcher	fant bindeleddet kivy	kivy sin apk-launcher fungerte dårlg, wsl fungerte dårlig	bytte til annen kompliator for kivy (buildozer) og sette opp linux	sette opp linux med buildozer
41	installert kivy på vm, rydda opp disk for installering, starta på linux dual boot	prøver å få buildozer til å fungere	fikk ikke koblet til tlf på vm, problemer med dual boot partisjonering pga. displass og windows	bruke dual boot istedenfor vm (allerede startet på), fortsette å fikse dual boot	installere linux, kivy, buildozer
42	installert linux dual boot og starta med kivy- og buildozer-installasjon, partisjonering for diskplass i linux	prøver å få buildozer til å fungere	partisjonering, os-kræsjing antakelig pga. diskplass	fikse kræsj og partisjonere mer diskplass	fikse linux, fortsette med kivy og buildozer
43	reinstallert alt	kivy hello world fungerer (python-kode på tlf)	ingen	ingen	muligens integrere maskinlæringskoden inn i med kivy-app
44	importere/formatere datasett	kivy hello world fungerer (python-kode på tlf)	bug ang. innlastning av datasett på forskjellig format	debugging/annen metode	fikse problemet
45	fortsatt med app-integrering	feil med opencv i buildozer, får ikke lenger kjørt hello world	feil med opencv (64 bit istedenfor 32 bit), testa med forskjellige konfigurasjoner	debugging buildozer/opencv/opencv-recipies	fikse problemet
46	app-integrering, reinstallerte buildozer	feil med opencv i buildozer, får ikke lenger kjørt hello world	feil med opencv (64 bit istedenfor 32 bit), testa med forskjellige konfigurasjoner	debugging buildozer/opencv/opencv-recipies	fikse problemet
47	fiksa buggen (reinstallere .buildozer-mappe), lagt til datasett, implementert accuracy-funksjon	trening av datasett kjører på tlf	kræsjer ved åpning av video (cv2.VideoCapture())	endre log-level, listdir for å sjekke om video eksisterer, prøve med kamera istedenfor video som input	fikse problemet
48	debugging med åpning av video/kamerafant ut at traceback mangler kun i dette tilfellet ved å lese full log fra telefonen rundt tiden kræsjen skjer	kjører på tlf, men kræsjer ved åpning av video	kræsjer ved åpning av video (cv2.VideoCapture(), får ikke traceback	utforske andre løsninger	fikse problemet
49	prøvd andre kameraløsninger (opencv, plyer, gstreamer, vanlig kamera-kall...)	kjører på tlf, men kræsjer ved åpning av video	får ikke åpna video/kamera, men vet at traceback mangler kun i dette tilfellet	utforske andre løsninger	fikse problemet
50	dokumentasjon	kjører på tlf, men kræsjer ved åpning av video	får ikke åpna video/kamera	Kaste inn håndkleet	dokumentasjon
51	dokumentasjon, presentasjon	kjører på tlf, men kræsjer ved åpning av video	får ikke åpna video/kamera		

Vedlegg III - Refleksjonsnotat Thomas

Gjennomføringen av prosjektet har vært en ekstremt lærerik prosess. Vi har jobbet med mange forskjellige teknologier, og har virkelig fått satt oss inn i avhengigheter på tvers av teknologier og operativsystemer. I tillegg har vi fått masse erfaring med utviklingsprosessen vi tok i bruk.

Samarbeidet mellom meg og Trym har vært godt. Vi har møttes ansikt-til-ansikt nesten hver gang vi har jobbet med prosjektet, noe som har ført til god og effektiv kommunikasjon. I tillegg har vi fått erfaring med parprogrammering, noe jeg likte godt.

Alt i alt er jeg svært fornøyd med både min egen og Tryms innsats, selv om resultatet ikke ble som ønsket. Vi har jobbet hardt, og prøvd så godt vi har kunnet hele veien. Jeg er veldig misfornøyd med valget av programmeringsspråk, da dette førte til at resultatet ble som det ble, men det er lett å si i etterkant. Ut ifra det vi fant på nett, virker kombinasjonen med kivy, python for android, og buildozer som en fin løsning. Problemene lå i de spesifikke versjonene og avhengighetene, noe som uansett ville vært vanskelig å forutsi før vi prøvde.

Takk til Alexander Holt som har veiledet oss og kommet med gode tips.

Vedlegg IV - Refleksjonsnotat Trym

Som det siste systemutviklingsprosjektet før bacheloroppgaven, synes jeg dette har vært veldig lærerikt og forberedende. Prosjektet var faglig nytt, i og med at vi fikk jobbe både apputvikling og maskinlæring, noe jeg aldri har hatt som prosjekt. Siden vi kun var to på gruppa, har jeg har lært å ta større ansvar for oppgaver, og å jobbe mer selvstendig enn med tidligere prosjekt. Dette gjorde at jeg fikk utfordret meg selv mer enn tidligere der jeg utgjorde en mindre andel av teamet.

Med Thomas på laget, synes jeg vi utgjorde en god duo, der samarbeidet var helt likestilt, og oppmøtet var det samme. Dette har gitt meg et litt annet syn på hvordan man kan jobbe effektivt sammen. Allikevel synes jeg det var kjipt å ikke nå opp til forventningene, når vi begge hadde ambisjoner om det. Oppgaven var vårt 1.-valg, og har tross alt vært spennende og “cutting edge” å jobbe med. Jeg vil også gjerne takke Alexander Holt, for et godt samarbeid og gode tips gjennom prosjektet.