



INSTITUTT FOR DATATEKNOLOGI OG INFORMATIKK

IDATT2501 - FORDYPNINGSPROSJEKT

---

## Klassifisering av trafikkskilt i sanntid med OpenCV i Android

---

*Forfattere:*

Simon Jensen, Stian Fjærana Mogen, Lars Brodin Østby

Desember, 2021

---

## Forord

Denne rapporten er skrevet av Simon Jensen, Stian Fjærå Mogen og Lars Brodin Østby. Oppgaven er gitt av Alexander Holt på vegne av 3D Motion Technologies. Hensikten med oppgaven var å undersøke maskinlæring for mobile enheter, og å finne ut i hvilken grad det var gjennomførbart å lage en mobilapplikasjon for sanntidsklassifikasjon av trafikkskilt.

For gruppen var dette en utmerket anledning til å fordype seg i og jobbe tverrfaglig med maskinlæring og applikasjonsutvikling. Samtidig er dette en unik mulighet for å prøve seg på en aktuell problemstilling, som bare kommer til å bli mer relevant i tiden som kommer.

Prosjektet har vært innholdsrikt og utfordrende, samtidig som givende og spennende. Gruppen har fått rikelig mengder kunnskap og erfaringer, og det er enighet at disse kommer til å komme godt med i fremtidige prosjektarbeid. Til tross for utfordringer, sitter gruppen igjen med motivasjon og iver for å ta fatt på bacheloroppgaven kommende semester.

Takksigelser utrekkes til veileder og oppdragsgiver Alexander Holt, som gjennom prosjektarbeidet har gitt verdigfull tilbakemeldinger. Samtidig har åpenheten for at gruppen selv kunne komme med egne betraktninger og innspill til oppgaven vært et friskt pust, og bidratt til selvstendighet i prosjektarbeidet.

Simon Jensen

Stian Fjærå Mogen

Lars Brodin Østby

---

## Oppgavetekst

Oppgaven går ut på å lage en applikasjon på Android (Java eller Kotlin), hvor man har en telefon på oppen på dashboardet i en bil og bruker kameraet til å kontinuerlig filme veien og tolke trafikkskilte (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 trenet opp et nevrat nett; vi koncentrerer oss i første omgang kun om fartsgrenser og vil utvide etter hvert. Det er en fordel om studenten(e) har tilgang til egen bil, men ikke et krav, og vi vil være behjelplig med å skaffe rådata til oppretting av modellen dersom det som finnes åpent ute på nett ikke er tilstrekkelig. Bruksområdene til en slik applikasjon er nærmest uendelige, men i første omgang er det et håp om kunne bruke data til å bidra til å føre opp skilt/bokser i OpenStreetMap, som er et crowd sourced alternativ til løsninger som Google Maps. Bare det å kunne vite fartsgrensen når en bruker OpenStreetMap vil være til stor hjelp.

Den originale oppgaveteksten har holdt seg ganske stabilt gjennom hele prosjektarbeidet. Det ble presisert tidlig at poengene som ble definert under bruksområder var forslag til hvordan resultatene fra oppgaven kunne brukes dersom man kom i mål med prosjektet. Samtidig ble det presisert tidlig at oppgavens kan bli sett på som et forskningsprosjekt. Hvorvidt prosjektet er gjennomførbart gitt gruppens ressurser er usikkert, dette beskrevet i visjonsdokumentet (siter her). Resultatene og betraktingene fra prosjektarbeidet vil derfor kunne være nyttige og ha en egenverdi uavhengig av om prosjektets sluttprodukt blir en suksess eller ikke. Dokumentasjon av prosessen, samt en detaljert beskrivelse av utfordringer og løsninger vil derfor også bli vektlegges i denne rapporten.

## Sammendrag

Med den raske utviklingen av mobile enheters hardware, er dens mulige bruksområder i maskinlæring et naturlig forskningsområde. Innsamling av data er en kritisk og ressurskrevende oppgave. Tilgjengeligheten av svært kapabelt hardware er nå større enn noensinne takket være mobiltelefonen. Dette skaper et enormt potensiale for at hvem som helst kan benytte seg av maskinlæring. Innsamling av vegdata er et eksempel på en disiplin som lenge kun har vært egnet for de store ressurssterke. Dersom hvem som helst kan ha en applikasjon på sin telefon som i sanntid detekterer og klassifiserer skilter ute på en kjøretur, vil selv en begrenset brukerbase kunne samle inn verdifulle data til bruk i open source prosjekter.

Applikasjonen som er laget i dette prosjektet er et bevis på at det er mulig å på en tilfredsstillende måte implementere maskinlæringsteknologi på mobile enheter, som er tilgjengelig for hvem som helst. Applikasjonen er en native Android applikasjon, og kan bli kjørt på alle nyere Android mobiltelefoner. Ved å bruke telefonens kamera fanger applikasjon opp fartsgrenseskilt den ser langs veien, og gir i sanntid tilbakemelding til brukeren om hvilken fartsgrense den har registrert. Sammen med prediksjonen, registrerer den enhetens lokasjon ved deteksjon. Brukeren vil i etterkant ha mulighet til å gå gjennom og verifisere alle trafikkskilte som applikasjonen har registrert og klassifisert, i et eget brukergrensesnitt.

Sanntidsprediksjonen gjøres ved hjelp av OpenCV. Maskinlæringsmodellen er laget i Pytorch, mens deteksjonen blir gjort med en Haar Cascade fil. Rapporten gir et bilde på hvilke utfordringer som dukker opp når en utvikler en mobilapplikasjon som benytter seg av maskinlæringsteknologi. Det blir også gjort rede for valgene av teknologier som er gjort i løpet av prosjektet, samt hvilke styrker og begrensninger disse har ført med seg.

Resultatet av dette prosjektet skal ikke bare være et nyvinnende produkt, men skal også gi innsikt i nyttig forskning som andre fagfolk kan benytte seg av i egne prosjekter. Det skal både være mulig å bygge videre på dette prosjektet, samtidig som at betraktingene som er gjort i løpet av prosessen skal være verdifulle for de som eventuelt leser dem.

---

# Innhold

<b>1</b>	<b>Introduksjon og relevans</b>	<b>1</b>
1.1	Akronymer og forkortelser . . . . .	1
1.2	Definisjoner . . . . .	1
<b>2</b>	<b>Teori</b>	<b>1</b>
2.1	Maskinlæring . . . . .	1
2.2	Sanntidsdeteksjon . . . . .	2
2.3	Konvolusjonelt Nevralt Nettverk . . . . .	2
2.4	Contrast Limited Adaptive Histogram Equalization . . . . .	2
2.5	Spatial Transformer Networks . . . . .	3
<b>3</b>	<b>Valg av teknologi og metode</b>	<b>3</b>
3.1	Teknologier . . . . .	3
3.1.1	Android Applikasjon . . . . .	3
3.1.2	Maskinlæring . . . . .	3
3.2	Metode . . . . .	4
3.2.1	Prosess . . . . .	4
3.2.2	Datasett og forarbeid . . . . .	5
3.2.3	Modellering og applikasjon . . . . .	5
3.2.4	Sanntidsdeteksjon . . . . .	5
3.2.5	Forbedringer og problemløsninger . . . . .	6
3.3	Arbeids- og rollefordeling . . . . .	9
<b>4</b>	<b>Resultater</b>	<b>10</b>
4.1	Vitenskapelige resultater . . . . .	10
4.1.1	Arktitektur . . . . .	10
4.1.2	Presisjon og tap . . . . .	10
4.1.3	Forvirringsmatrise . . . . .	12
4.1.4	Utvidelse av datasettet . . . . .	13
4.1.5	Sanntidsdeteksjon . . . . .	14
4.1.6	Android Applikasjon . . . . .	14
4.2	Ingeniørfaglige resultater . . . . .	16
4.2.1	Produktets funksjonelle egenskaper . . . . .	16
4.2.2	Ikke funksjonelle egenskaper og andre krav . . . . .	17
4.3	Administrative resultater . . . . .	17

---

4.3.1	Fremdriftsplan . . . . .	17
4.3.2	Tidsforbruk . . . . .	18
<b>5</b>	<b>Diskusjon</b>	<b>18</b>
5.1	Datasett . . . . .	18
5.2	Maskinlæringsmodell . . . . .	19
5.3	Android Applikasjon . . . . .	20
5.4	Sanntidsprediksjon . . . . .	20
5.5	Samarbeid mellom teknologier . . . . .	21
5.6	Prosjektarbeidet . . . . .	21
<b>6</b>	<b>Konklusjon</b>	<b>22</b>

## Figurer

1	Skilt fra testsett . . . . .	7
2	Skilt fra app . . . . .	7
3	120 skilt . . . . .	7
4	Transformerte skilt . . . . .	8
5	Distribusjon originalt . . . . .	8
6	Distribusjon etter transformering . . . . .	8
7	Bilder uten STN . . . . .	9
8	Bilder med STN . . . . .	9
9	CNN arkitektur . . . . .	10
10	Presisjon treining og validering . . . . .	11
11	Tap treining og validering . . . . .	11
12	Forvirringsmatrise . . . . .	12
13	Presisjon treining og validering . . . . .	13
14	Tap treining og validering . . . . .	13
15	Kameraaktivitet . . . . .	15
16	Valideringsaktivitet fremvisning . . . . .	15
17	Valideringsaktivitet oppdatere skilt . . . . .	15
18	Timefordeling . . . . .	31

## Tabeller

1	Resultater epoke 1-4 . . . . .	11
---	--------------------------------	----

---

2	Resultater epoke 26-29 . . . . .	11
3	Presisjon uavhengig testsett . . . . .	12
4	Sammenligning av presisjon på test-, validering og treningssett . . . . .	12
5	Resultater utvidet, epoke 1-4 . . . . .	13
6	Resultater utvited, epoke 13-16 . . . . .	13
7	Sammenligning av presisjon på test-, validering og treningssett . . . . .	14
8	Resultater fra cascade classifier på treningsdata . . . . .	14

---

# 1 Introduksjon og relevans

Dette prosjektet har som formål å utforske i hvor stor grad det er mulig å benytte mobil teknologi til å utføre deteksjon og tolkning av trafikkskilt i sanntid. Prosjektet håper å utforske både mulighetene og begrensningene til en mobil enhets maskinvare, samt mulighetene til å utvikle tilstrekkelig programvare til å utføre oppgaven på et tilfredsstillende nivå. Et underliggende mål ligger i en muligheten for at prosjektets resultat i fremtidig arbeid kan bli brukt til dugnad i innsamling av data om trafikkskilt. Informasjonen som blir samlet inn skal i fremtiden kunne bli sendt til et åpent kildekode kart api. Et slikt dugnadsprosjekt vil igjen la utviklere ta nytte av den frie informasjonen i utvikling av nye prosjekter.

Denne rapporten skal gi en fullstendig oversikt over gruppens prosjektarbeidet. Den tar for seg relevant teori, gruppens valg av teknologi og metode, samt presentere prosjektets vitenskapelige, ingeniørfaglige og administrative resultater. Videre innbefatter rapporten en diskusjonsdel hvor resultatene blir drøftet. Leseren får her et innblikk i årsakene til at resultatene ble som de ble, og hvilke avvik fra den originale problemstillingen og mål som oppstod. Avslutningsvis tar rapporten for seg en konklusjon av prosjektarbeidet og det oppnådde resultatet, samt muligheter for fremtidig utvikling av prosjektet.

## 1.1 Akronymer og forkortelser

- CNN : Konvolusjonalt nevralt nettverk
- YOLO: You Only Look Once
- ONNX: Open Neural Network Exchange
- CLAHE: Contrast Limited Adaptive Histogram Equalization
- STN: Spatial Transformer Networks

## 1.2 Definisjoner

- Sanntid: Tilnærmet øyeblikkelig respons
- Deteksjon: Oppdagelse av input for prediksjon
- Prediksjon: Klassifikasjonen som en maskinlæringsmodell gjør basert på input

---

# 2 Teori

## 2.1 Maskinlæring

Maskinlæring er en undergruppe av kunstig intelligens. Maskinlæring er studiet av å gjøre maskiner mer menneskelignende i sin oppførsel og beslutninger ved å gi den muligheten til å lære og utvikle sine egne programmer. Dette gjøres med et minimum av menneskelig inngrisen, noe som vil si ingen eksplisitt programmering. Læringsprosessen automatiseres og forbedres basert på maskinenes erfaringer gjennom hele prosessen. Data av god kvalitet mates til maskinen, og ulike algoritmer benyttes til å bygge maskinlærings modeller for å trenne maskinene på data den blir matet. Valget av algoritme avhenger av hensikten bak modellen, altså typen aktivitet som må automatiseres, samt hvilken datatype man trener modellen på.[14]

---

## 2.2 Sanntidsdeteksjon

Sanntidsdeteksjon av objekter er oppgaven av å utføre gjenstands gjenkjenning og identifisering i sanntid med rask slutning, samtidig som et grunnleggende presisjonsnivå blir opprettholdt. Sanntidsdeteksjon kan ofte være en prosess hvor det detekteres et objekt man søker etter og predikerer det direkte. Men i tilfeller hvor man ønsker å kunne predikere flere ulike typer objekter som kan være svært like kan man dele prosessen i to. Her vil man først detektere et objekt av interesse, for å så predikere eksakt hvilket av de mulige objektene man søker etter dette objektet er. En enhet sitt kamera kan bli brukt til å detektere objekter i et miljø ved hjelp av maskinlæring. [3]

## 2.3 Konvolusjonelt Nevralt Nettverk

Et konvolusjonalt nevralt nettverk (CNN) er en dyp læring algoritme som kan ta inn et inngangsbilde (input), tildele betydning (lærbare vekter og skjeheter (weight bias)) til de ulike elementene og objekter den finner i bildet, samt være i stand til å skille disse ulike elementene fra hverandre. CNN brukes ofte til klassifisering, data-synsoppgaver (computer vision) [2] og analyse av bilder.

CNN er en type nevralt nettverk. Generelt er nevralt nettverk en undergruppe av maskinlæring, og ligger i kjernen for dype lærings algoritmer. Nettverkene består av en rekke nodelag, som inneholder et inngangslag (input layer), ett eller flere skjulte lag (hidden layer) og et utgangslag (output layer). Hver node kobles til en annen, og har en tilhørende vekt og terskelverdi. Hvis utgangen fra en individuell node når over denne angitte terskelverdien vil den gitte noden aktiveres, og sendes data til neste lag i nettverket. Hvis ikke denne terskelverdien oppnås vil ingen data bli sendt videre til neste lag i nettverket.

Konvolusjonelle nevrale nettverk skiller seg fra andre nevrale nettverk ved deres overlegne ytelse med å tolke bilde-, tale- eller lydsignaler. I et konvolusjonalt nevralt nettverk finner man hovedsakelig tre hovedtyper av lag, som er:

- Konvolusjons lag (Convolutional layer)
- Samlings lag (Pooling layer)
- Fullt tilkoblet lag (Fully-connected layer)

Konvolusjons er det første laget i nettverket. Et konvolusjonslag kan enten følges av ytterligere konvolusjons lag, eller sammenslåingslag. Uavhengig av det er det til slutt det fullstendig sammenkoblede laget som er det siste laget i nettverket. Med hvert lag øker CNN sin kompleksitet, og har mulighet for å identifisere større deler av et gitt bilde.

De tidlige lagene i nettverket fokuserer på enkle elementer av et bilde, som farger eller kanter. Etter hvert som bilde dataene går gjennom lagene til nettverket, begynner den å gjenkjenne større elementer eller former av objektet til det endelig identifiserer det tiltenkte objektet.[13]

## 2.4 Contrast Limited Adaptive Histogram Equalization

CLAHE er en teknikk som benyttes innen bildebehandling. Teknikken bruker histogram equalization for å forbedre kontrastnivået i bildet. I histogram equalization strekkes histogrammet for å inkludere alle verdier. Dersom et bilde kun har røde fargeverdier mellom 60 og 70, strekkes dette ut slik at de laveste verdiene får verdi 0 og de høyeste verdiene blir 255. CLAHE benytter denne teknikken på ulike isolerte deler av bildet. Dette gjør den for at veldig lyse og mørke regioner ulike steder i bildet ikke ødelegger for Histogram Equalization hos hverandre. [6]

---

## 2.5 Spatial Transformer Networks

Et problem med Convolutional Neural Networks (CNN) er at det er vanskelig å lære små variasjoner i bildet, for eksempel forskyvninger eller rotasjoner, på en effektiv måte. For noen år siden kom DeepMind ut med forskning som gjorde det mer effektivt å trenere opp modellen for å gjenkjenne små varianser. Dette gjorde de gjennom Spatial Transformer Networks (STN). Spatial transformer nettverk består av tre deler, et lokaliseringsnettverk, en grid generator og en sampler. Lokaliseringensnettverket predikerer hvilken transformasjon som skal gjøres. Den sender output parametre av transformasjonen videre til grid generatoren. Her lages en sampling grid, som er en mapping av de ulike input punktene. Denne bruket til slutt av sampleren for å utføre transformasjonen. En fordel med STN er at det er enkelt å implementere i en ferdiglaget modell uten store modifikasjoner. [4]

# 3 Valg av teknologi og metode

## 3.1 Teknologier

### 3.1.1 Android Applikasjon

Visjonsdokumentet (Vedlegg 6) beskriver et behov for å kunne bruke OpenCV for sanntids deteksjon i en mobilapplikasjon. Fordypningsprosjektet ønsker å bygge på og videreføre kunnskap som gruppemedlemmene opparbeider seg i sine respektive emner. To av gruppens tre medlemmer tar emnet INFT2501 Applikasjonsutvikling for mobile enheter parallelt med fordypningsemnet. Å lage applikasjonen som en Android applikasjon er hensiktsmessig av flere grunner. Først og fremst er Android opensource [3], og gruppens overordnede vurdering er at prosessen med å gjennomføre et maskinlæringsprosjekt med OpenCV er mest gjennomførbart i Android. Problemstillingen beskriver fremtidige utvidelser til å være integrasjon mot open source kart api.

#### Native Android

En applikasjon i Android kan skrives på flere forskjellige måter. En kan skrive en Native Applikasjon, i Java eller Kotlin. Man kan også skrive en kryssplattformløsning som fungerer på både Android og iOS. Fordelen med native Android er at dette støtter OpenCV, samtidig som at den støtter en mengde maskinlæringsbiblioteker som Pytorch Mobile og TensorFlow Lite. Disse alternativene er svært godt dokumentert. Da prosjektet i stor grad kan anses som et forskningsprosjekt med behov for å utforske forskjellige teknologier og veie disse opp mot hverandre, blir derfor native Android et naturlig valg. Av samme grunn blir derfor Java foretrukket over Kotlin, da Java er mer etablert med større grad av dokumentasjon opp mot OpenCV. [11] Det skal likevel nevnes at det endelige valget om å gå for Java fremfor Kotlin ble tatt noen uker inn i prosjektet, etter at gruppen jobbet parallelt med begge språk. Det viste seg her at tilgjengelig nettressurser og tidligere arbeid i Java gjorde det overlegen for gruppen sitt tilfellet.

#### Android Studio

Det valgte utviklingsmiljøet for applikasjonen er Android Studio. God støtte for både emulator og testing med egen enhet gjør Android Studio til et naturlig valg. Android Studio sin debugger og generelt mange muligheter gjør det til utviklingsmiljøet som i størst grad oppfyller gruppens behov for gjennomføring av prosjektet. Støtte for enkel importering av diverse teknologier veide her også stort, dette viste seg derimot å likevel by på utfordringer som blir presentert senere i rapporten.

### 3.1.2 Maskinlæring

Maskinlæring er et bredt begrep, og mulige anvendelser av maskinlæring er uendelige. For å kunne gjennomføre et prosjekt som dette er man avhengig av å kunne ta informerte valg knyttet opp mot relevant teori. Samtidig gir valg av miljø og resultatmål enkelte begrensninger og utfordringer hvor løsningen ikke blir åpenbar før man har opparbeidet seg relevant kunnskap samt prøvet og feilet.

---

## OpenCV

Å predikere samt detektere i sanntid byr på ekstra utfordringer i klassifikasjonsproblemet. OpenCV er et open source bibliotek som er støtter flere forskjellige språk, inkludert Java, og var nevnt som et krav for gjennomføringen av prosjektet. OpenCV er et naturlig valg for bildeprosessering i sanntid. Både på grunn av støtten til native Android applikasjoner, men også som en konsekvens av at det er svært veletablert og dokumentert innenfor maskinlæringsmiljøet.

### Haar Cascade

I maskinlæringsoppgaver om objektgjenkjenning er det vanlig å dele oppgaven i to deler. Den første delen av oppgaven er å finne ut hvor i bildet objektet befinner seg. Det er flere verktøy som kan løse dette problemet, der YOLOv3 er et populært valg [5]. Noe av problemet med denne i forbindelse med utvikling på mobil er derimot at det er beregningsmessig tungt å kjøre. Vi valgte derfor bruke en HaarCascade classifier. [16]

HaarCascade (også kalt Harr klassifiserer) er et maskinlærings objekt som brukes for objekt deteksjon. Når OpenCV leser inn bilder fra en video i sanntid er det Haar Cascade som analyserer bildet, og returnerer x- og y koordinater til deler av bilde som treffer visse mønstre. I 2001 utga P.Viola og M.Jones ut artikkelen “Rapid object detection using boosted cascade of simple features”. [15] Dette ble introduksjonen til sanntidsdeteksjon ved hjelp av cascade funksjon. Denne er trent opp ved hjelp av mange positive bilder (bilder av objektet den skal finne) og negative bilder (bilder uten objektet den skal finne). [10]

### PyTorch

PyTorch er et svært populært bibliotek for å lage maskinlæring modeller i Python. Biblioteket er svært fleksibelt, og fasiliteter for dyp-læring av modeller. Pytorch har også CUDA støtte som gir den muligheten til å benytte seg av GPU'en under trening, gitt at maskinen som trener modellen har en NVIDIA GPU.

## 3.2 Metode

### 3.2.1 Prosess

Som konsekvens av at prosjektet bærer forskningspreg, er det ikke lett å bestemme fastsatte datoer i planleggingsfasen. Derimot var det viktig for gruppen å ha en klar plan med tydelige målsetninger, som naturligvis vil kunne justeres etter behov. Dette ble utarbeidet sammen med veileder Alexander Holt, hvor det også ble avtalt å ha en MVP klar innen slutten av oktober. Basert på dette lagde gruppen en fremdriftsplan.

#### 1 Datasett og forarbeid

Denne fasen gikk også i stor grad ut på å tilegne seg kunnskap om aktuelle teknologier. Samtidig begynne med arbeidet å finne egnet datasett.

#### 2 Modellering og applikasjon

Begynne med arbeid av modellen, samt å lage en enkel kameraapplikasjon.

#### 3 Sanntidsdeteksjon (Ferdig 31. oktober)

Predikere trafikksilt med OpenCV i applikasjon, samt detektere trafikkskilt i sanntid. Dette var tilstrekkelig for en prototype.

#### 4 Forbidering (Ferdig slutten av desember)

Gjøre modellen og implementasjonen av maskinlæringsmetodene tilfredstillende i Android Applikasjonen. Preprosessering hører til i denne fasen.

#### 5 Ferdigstilling (Siste frist 3.januar) Rapportskriving og eventuelle funksjonelle utvidelser og bugfiksing.

---

### **3.2.2 Datasett og forarbeid**

En av de største utfordringene gruppen hadde under hele prosjektet var å finne tilstrekkelig treningsdata som holdt mål både i kvalitet og kvantitet på dataen. Det første datasettet som ble prosessert og som skulle bli brukt til trening ble senere lagt til side for å være for lite da settet består av 600 bilder totalt fordelt på alle klassene, når ønskelig mengde er å ha i hvert fall mange hundre bilder av hver eneste klasse. I stedet for ble heller German Traffic Sign Recognition Benchmark (GTSRB) tatt i bruk. Dette var det største offentlige datasettet gruppen fant som inneholdt fartsgrenseskilt. [8] Datasettet består av om lag 36 000 bilder hvorav 17500 er av fartsgrenser. Vi valgte først å kun bruke fartsgrense delen av datasettet. Dette var da bilder av fartsgrenser fra klassene 20, 30, 50, 60, 70, 80, 100, 120 og oppheving av 80.

### **3.2.3 Modellering og applikasjon**

Gruppen avgjorde allerede i planleggingsfasen, at det var nødvendig å starte med selve applikasjonen tidlig i prosjektet. Denne avgjørelsen ble tatt på bakgrunn av tidligere erfaringer med nye teknologier som gir uforutsette problemstillinger. For selve applikasjonen var det to konkrete problemstillinger som måtte løses i denne fasen. Den første og antatt enkleste var å lage en enkel kameraapplikasjon. Den andre var å bruke OpenCV sammen med en maskinlæringsmodell i Android.

#### **OpenCV i Android**

I nyeste versjonen av Android Studio Arctic Fox eksisterer det en bug som gjør det umulig å importere moduler slik som OpenCV direkte i IDE. Dette er blitt rapportert flere ganger, blant annet i dette issuet. [1] Alternativer løsninger på dette som har blitt foreslått er å importere direkte i gradle, eller bruke en tidligere versjon av Android Studio. Direkte inkludering i settings gradle fungerte ikke i gruppens spesifikke tilfelle, og å bruke en tidligere versjon ga etter noe testing ikke tilfredsstillende resultater. Løsningen gruppen kom frem til ble å laste ned et eksisterende tutorialprosjekt som allerede hadde modulene lagt inn. Dette ble laget av Adrien Lescourt. [7] Eksempelprosjektet fra 2019 var gammelt nok til at vi kunne jobbe rundt buggen i Android Studio, og brukte OpenCV versjon 3.4.1 som virket å være godt dokumentert. På denne måten ble denne løsningen tilsvarende resultatet gruppen ellers ville fått dersom importering av OpenCV modulene hadde lett seg gjøre på ordinært vis. Det er sannsynlig at dette spesifikke problemet løses i nærmeste fremtid, dette er derimot et eksempel på en typisk problemstilling i et slikt type prosjekt.

#### **PyTorch i Android**

For å importere modellen i Android, brukes Pytorch sin module. Dette gjør at vi kan laste inn modellen, samtidig som vi kan gjøre om vektorer til tensorer. Implementasjon av Pytorch i Android ga noe lignende utfordringer som ved implementeringen av OpenCV, dog ikke like problematisk.

Før Pytorch ble det endelige valget for å benytte seg av maskinlæringsmodellen i Android så gruppen på hvilke ulike alternativer man kunne utforske for implementeringen av modellen. Rammeverket ONNX (Open Neural Network Exchange) ble blant annet sett på. ONNX er et format rammeverk som brukes for å konvertere en maskinlæringsmodell mellom ulike rammeverk, for eksempel fra en modell i Pytorch rammeverket til TFLite rammeverket. Fordelen med ONNX er fleksibiliteten det gir å kunne implementere ulike modeller fra et rammeverk til et nytt et. Men beslutningen falt på å bruke Pytorch modulen til Android ettersom maskinlæringsmodellen allerede var bygd i Pytorch rammeverket, og det derav fantes ikke et behov for et mellomledd som konverterer formatet til modellen.[9]

### **3.2.4 Sanntidsdeteksjon**

#### **Haar Cascade**

For objekt deteksjon ble et Haar Cascade objekt implementert. Det finnes andre alternativer til

---

Haar Cascade til bruk av objekt deteksjon. En av de mest populære er av YOLO (You Only Look Once) algoritmene. Det finnes flere ulike YOLO varianter, men en generell forskjell mellom YOLO og Harr Cascade er at Harr Cascade er en “en objektklasse” deteksjons algoritme. Mens YOLO kan detektere flere ulike klasser av objekter forholder Haar Cascade seg best til bare en klasse. Dette gjør YOLO igjen mer kompleks enn Haar Cascade, og mer tungvint å implementere i Android. Ettersom det i hovedsak for dette prosjektet dreide det om en objektklasse i form av fartsgrense-skilt ble Haar Cascade sett på som det bedre valget.

Haar Cascade må som andre maskinlærings objekter bli trent opp med tilstrekkelig treningsdata for å yte godt. Den trener opp modellen og lagrer kjennetegn ved objektene i en xml fil. Denne brukes deretter i sanntids bildetakingen for å lete etter former som kan matche trekene. Ettersom mangelen på god og store nok mengder treningsdata var en stor utfordring gruppen støtte på under prosjektets løp, valgte gruppen å benytte seg av en ferdigtrent Haar Cascade modell som ble funnet på nett.[12] Gruppen ga et forsøk på å finne positive og negative bilder til å trenere opp en modell for hånd. Det ble samlet inn 450 positive bilder og 2000 negative, og trentes opp gjennom opencv. Det vi derimot fort merket ved bruk av denne xml filen var flere falske positive objekter funnet under testing på reelle bilder og videoer. Det ble derfor ikke lagt mye mer tid inn i dette, ettersom innsamling av bilder var svært tidkrevende. Dette frigjorde mye av gruppens tid da man slapp å lage et stort nok datasett selv, men det gikk på bekostning av objekt deteksjonen i applikasjonen ikke er så presis som gruppen skulle ønsket. Dette kommer av at modellen som blir tatt i bruk ikke har tilfredsstillende høy presisjon.

#### Prototype

Prototypen som gruppen satt igjen med på møtet var et minimum valuable product som sterkt tydet på at produktet var gjennomførbart, men som hadde kritiske mangler før det kunne tas i bruk. En prototype sin funksjon er først og fremst for å sette klare målsetninger for gruppen å jobbe etter, samt å gi interessenter noe håndfast på et visst tidspunkt i utviklingsløpet. Det er likevel strengt nødvendig å presentere observasjoner og betraktninger fra prototypen allerede nå i rapporten. Både som følge av forskningspreget til prosjektoppgaven, men også som en konsekvens av de interessante og tilsynelatende selvmotsigende resultatene som gruppen fikk. Den siste fasen av prosjektet vil i aller høyeste grad basere seg på disse konkrete observasjonene.

Først og fremst er det verdt å nevne at gruppen innfridde på punktene som var satt opp som mål sammen med produkteier. På dette tidspunktet er Android Applikasjonen en enkel kamera applikasjon, som bruker en ferdig trenet maskinlæringsmodell for å predikere trafikkskilt i sanntid. Den umiddelbare svakheten til applikasjonen var at prediksjonene som ble gjort, nærmest virket å være vilkårlige. De eneste prediksjonene som tilsynelatende ble riktige mer enn et vilkårlig utvalg, var for 50km/h og 60km/h. Det var derimot fortsatt uklart hvorvidt dette kom som en konsekvens av at modellen ga prediksjoner basert på treningsdataen, eller om valgene som ble tatt i praksis var gjetning. Vi vet at den ferdigtrente modellen gir en presisjon på over 95 prosent på det uavhengige testsettet i et kontrollert testmiljø, hvor alle bildene er på et format vi vet modellen håndterer bra. Det prototypen har vist oss er at gruppen videre må se på hvorfor bildene som blitt tatt inn og predikert i Android applikasjonen gir så store avvik fra forventet atferd. Siste fase, og dermed resten av dette kapittelet vil ta utgangspunkt i forskningen og de tilhørende tiltakene som ble gjort for å gjøre applikasjonen fungerende basert på observasjonene gjort her.

#### 3.2.5 Forbedringer og problemløsninger

Basert på observasjonene gjort i forrige fase, var det noen kritiske spørsmål gruppen trengte svar på før noen form for progresjon kunne bli gjort. Den sentrale problemstillingen var hvorfor prediksjonene ble så forskjellig mellom applikasjonen og det kontrollerte testmiljøet. For å få svar på dette er det nødvendig å se på hvilke bilder applikasjonen sender til modellen, og se hvordan dette er forskjellig fra bildene fra trening- og testsett.



Figur 1: Skilt fra testsett

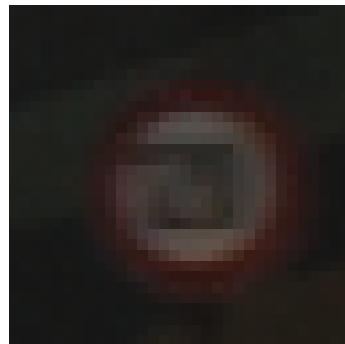


Figur 2: Skilt fra app

Figur 1 viser eksempel på et bilde som modellen trenes opp og blir testet på. Bildet i figur 2 viser hvordan et bildet ser ut når applikasjonen detekterer det og sender det til modellen for prediksjon. Vår intuisjon sier at bildet til høyre ikke bare skal være godt nok for prediksjon, men at det i tillegg er mer leselig enn bildene som modellen trener nå. Det er positivt å se at modellen faktisk bruker et leselig bildet, men det gir ikke et umiddelbart svar på hva som går galt. Det er derfor nødvendig å se videre på hva som potensielt kan skape forvirringen.

#### Augmentation

Det var flere problemer med datasettet. For det første var datasettet laget fra skjermutklipp fra etsekunds videosnutter. Mange av bildene var ganske uleselige selv for mennesker. Dette er illustrert i figur 3



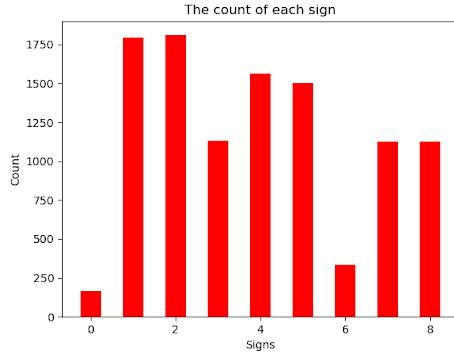
Figur 3: 120 skilt

Selv om datasettet består av om lag 17500 fartsgrenseskilt, er dette fortsatt ikke tilstrekkelig for å trenne en god, generell modell. Fordi bildene ble hentet fra videosnutter er bildene gruppevis ganske veldig like og under samme lysforhold. Dette problemet ble løst gjennom augmentering av bildene. Vi roterte, zoomet, vridde og tok tilfeldige utklipp av bildene for å skape større variasjon i datasettet. Dette ble gjort for å øke mulighetene for hva input modellen kunne predikere riktig. Dersom et skilt ved veien hadde veltet, og vært vridd 90 grader skal applikasjonen optimalt sett fortsatt kunne predikere skiltet.

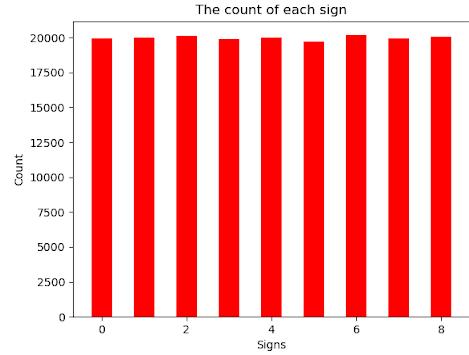


Figur 4: Tilfeldig transformerte skilt

Et annen problem i datasettet var distribusjon mellom de ulike klassene. Dette er illustrert på figure . Eksempelvis er det rundt 200 forekomster av 20-skilt, mens det er nesten 2000 30-skilt og 50-skilt. Dette gjorde at når modellen ble trent ville den oftere gjette de overrepresenterte klassene i forhold til de underrepresenterte, ettersom dette minket loss og økte presisjon. Dette problemet løste vi gjennom en vektet sampler. Denne sampleren tar inn bildene og vekter. Disse vektene beskriver hvor mange ganger dette bildet skal brukes. Hver gang den brukte bildene kjørte den tilfeldig augmentering slik at disse ikke skulle bli like. Vi valgte at det skulle totalt være omtrent 20 000 bilder av hver klasse.



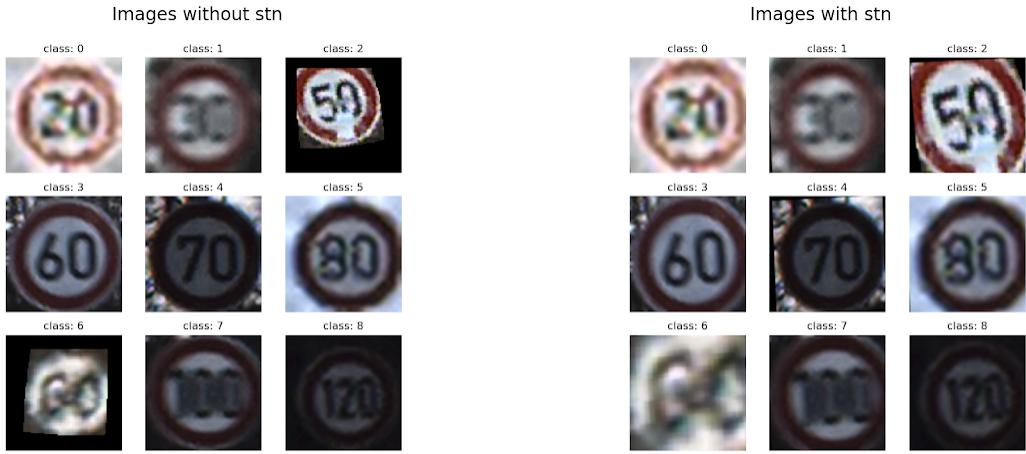
Figur 5: Distribusjon originalt



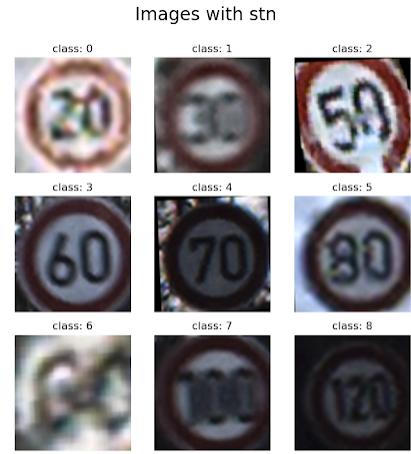
Figur 6: Distribusjon etter tranformering

### Spatial Transformer Network

For å gjøre læringen til modellen raskere og bedre implementerte vi et Spatial Transformer Network. Dette tar inn et bilde, lokaliserer hvor en transformasjon skal gjøres, genererer en transformasjons grid og utfører transformasjonen på bildet. Den skal under trening lære seg hvilke områder i bildene som er nyttige å inkludere i bildet og eventuelle “romlige” transformasjoner som måtte gjøres. Vi plasserte STN først i modellen slik at den skulle lære seg de tilfeldige transformasjonene som ble gjort på datasettet, og gi videre et mer standardisert bilde til CNN.



Figur 7: Bilder uten STN



Figur 8: Bilder med STN

### Bilbehåndtering og Geolokasjon

Visjonsdokumentet beskrev et behov for å kunne liste opp hvilke trafikkskilt som ble registrert, samt en mulighet til å bekrefte og avkrefte hvorvidt prediksjonene som ble gjort er korrekte. Det ble derfor implementert en enkel bilbehåndteringsaktivitet i Android applikasjonen. I det et bilde blir detektert, blir det også tildelt en lokasjon basert på den mobile enheten sin posisjon. Android har en innebygget LocationListener som kan implementeres i aktiviteten. Denne oppdaterer seg etterhvert som enheten sin lokasjon oppdaterer seg, og gir derfor en enkel måte å lagre en tilnærmet posisjon for trafikkskiltet. Som følge av at denne delen av applikasjonen var prioritert noe lavere i Visjonsdokumentet enn selve sanntidsprediksjonen, følte gruppen dette ga en adekvat løsning på problemstillingen.

### Klassifisering av flere trafikkskilt

Avslutningsvis ønsker vi å forske på hvordan arbeidet og metodene fungerte på det fullstendige datasettet. Dette vil si at man inkluderer alle trafikkskilt som kom med i GTSRB som tilsammen hadde 43 forskjellige klasser, en betraktelig økning fra de originale 9 som modellen hadde trent på. Allerede før dette forsøkes kan man med svært stor sikkerhet si at presisjonen på denne modellen blir lavere som en konsekvens av et økt antall klasser. Det er likevel svært interessant å se hvordan modellen vil klare seg med de samme parameterne. Dersom det viser seg at det er mulig å oppnå gode resultater også med et stort utvalg slike klasser, kan dette være svært verdifull informasjon som beviser at det er mulig å bygge videre og utvidet arbeidet som er gjort.

### 3.3 Arbeids- og rollefordeling

Prosjektarbeidet er et fordypningsprosjekt hvor gruppemedlemmene får mulighet til å gå i dybden på et eller flere av emnene som de tar parallelt med prosjektet. Det er to aktuelle emner som i denne sammenhengen er relevant for prosjektet. INFT2501 Applikasjonsutvikling for mobile enheter, og IDATT2502 Anvendt maskinlæring med prosjekt. Maskinlæringsfaget IDATT2502 er et emnet alle gruppens medlemmer har. INFT2501 har derimot kun Stian Fjærå Mogen og Lars Brodin Østby. Dette førte med seg naturlige konsekvenser for rollefordelinger.

Siden gruppen avgjorde å starte med den mobile applikasjonen og maskinlæringsmodellen parallelt, ble det naturlig for Simon Jensen og starte med maskinlæring. I oppstartsfasen var kun Mogen og Østby ansvarlig for applikasjonen. Dette gjaldt å sette opp prosjekt og å laste inn nødvendige moduler. Samtidig ble delene av applikasjonen som ikke var relatert til maskinlæring gjennomført av Mogen og Østby. Arbeidsfordelingen utviklet seg over prosjektet, som en konsekvens av at arbeidsoppgavene med applikasjonen endret seg. Etter hvert ble problemstillingene mer relatert til

---

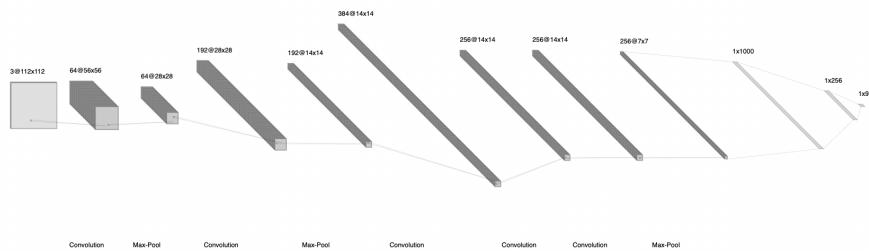
bruken av OpenCV og PyTorch bibliotekene, slik at alle igjen satt med samme forutsetningene for å utvikle den mobile applikasjonen.

Gruppen var klar over at det i aller høyeste grad kom til å bli nødvendig at alle gruppens hadde kompetanse om begge delene av prosjektet. Selv om oppstarten hadde en nødvendig rollefordeling, utviklet det seg likevel til et prosjektarbeid hvor denne ulikheten jevnet seg ut. Alle gruppemedlemmene var delaktige i samtlige aspekter av prosjektet, og uavhengig av ansvarsområder ble gruppens medlemmer informert om endringer og forbedringer. Dette ble gjort slik at aldri noen skulle sitte med en følelse av at de ikke visste hvilke arbeidsoppgaver som måtte gjøres.

## 4 Resultater

### 4.1 Vitenskapelige resultater

#### 4.1.1 Arktitektur

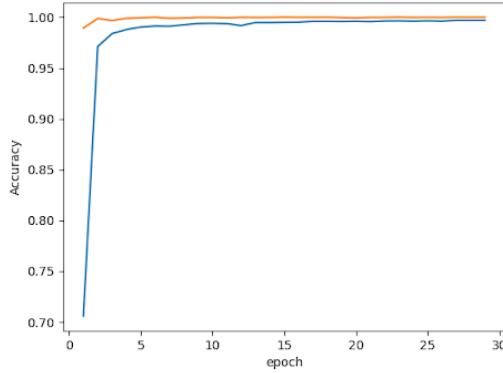


Figur 9: CNN arktitektur

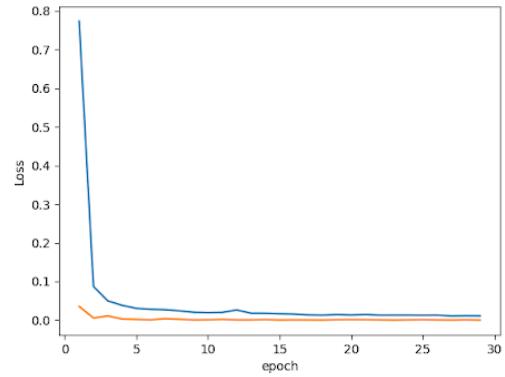
Vi endte opp med et konvolusjonelt nevralt netverk for selve identifikasjon av skiltene. Vi utviklet en modell som hadde flere konvolusjonelle lag. Mellom flere av lagene benyttet vi max pooling for å redusere størrelsen til modellen og samtidig redusere forskjeller i rotasjon forskyvning og skalering. Vi benyttet også regularisering metoden Dropout. Denne gjør at enkelte lærte hidden layer nevroner blir ignorert under trening, og forhindrer overfitting av modellen.

#### 4.1.2 Presisjon og tap

Under trening satt vi modellen til å gå igjennom treningsdata maksimalt 50 ganger, men at den skulle stoppe dersom resultat på valideringsdata ikke forbedret seg på 5 epoker. Vi lagrer også bare modeller som gir lavere loss eller høyere treffsikkerhet enn beste resultat. Dette gjorde at vi ikke trengte å vite nøyaktig hvor lenge vi skulle trenre modellen.



Figur 10: Presisjon trening og validering



Figur 11: Tap trening og validering

Figur 10 og figur 11 viser resultatene fra kjøringene på den endelige modellen som ga best resultater. Direkte fra grafen ser man at treningssettet (blått) begynner med mye lavere presisjon og mye høyere loss enn valideringssettet (oransje), som ser ut til å nå tilnærmet beste verdi allerede ved de første epokene. Dette er resultater som ikke er veldig intuitive, men de kan forklares. Årsaken til dette var at modellen var i treningsmodus under treningen mens den ble satt i evaluatingsmodus under test av valideringssettet. Dette innebar at på grunn av Dropout, som kun er aktiv under trening av modellen, hadde modellen tilgang på flere nevroner og mer prosessorkraft under vurdering av valideringssettet. Denne teorien etterprøvde vi gjennom å kjøre test settet én epoch i evaluatingsmodus og fikk treffsikkerhet på 0.9889, som var veldig likt validering.

Tabell 1: Resultater epoch 1-4

Epoch	Train acc	Valid acc	Train loss	Valid loss
1	0.7058	0.9894	0.7734	0.0353
2	0.9712	0.9987	0.0869	0.0055
3	0.9838	0.9967	0.0497	0.0106
4	0.9878	0.9989	0.0382	0.0030

*Presisjon og tap for trening- og valideringssett epoch 1-4*

Tabell 2: Resultater epoch 26-29

Epoch	Train acc	Valid acc	Train loss	Valid loss
26	0.9962	0.9998	0.0128	0.0005
27	0.9969	1.0000	0.0107	0.0001
28	0.9969	0.9999	0.0113	0.0008
29	0.9969	1.0000	0.0110	0.0001

*Presisjon og tap for trening- og valideringssett epoch 26-29*

De to tabellene gir en mer presis oversikt over nøyaktig hvor god presisjon og loss som ble oppnådd gjennom kjøringene. Som nevnt starter både presisjon og tap for valideringssettet svært nære de endelige resultatene, mens treningssettet bruker 3-4 epoker på å nå tilsvarende resultater. Valideringssettet stabiliserer seg på 100 prosent presisjon og ingen loss, som er litt høyere i presisjon og lavere i loss sammenlignet med treningssettet. Begge resultatene er svært høye, og som isolert sett kan være grunn til bekymring for at modellen overfitter. Det ble derfor testet på det uavhengige testsettet for hver epoch parallelt med validering og trening, for å forsikre om at de gode resultatene var reelle.

Tabell 3: Presisjon uavhengig testsett				
Epoke	26	27	28	29
Presisjon	0.9914	0.9879	0.9891	0.9914
<i>Uavhengig testsett</i>				

Tabellen viser at også testsettet gir presisjon over 99 prosent nøyaktighet. Siden settet er helt uavhengig fra både trenings- og valideringssettet bekrefter det også at modellen sine resultater ikke kommer av en feil i treningen.

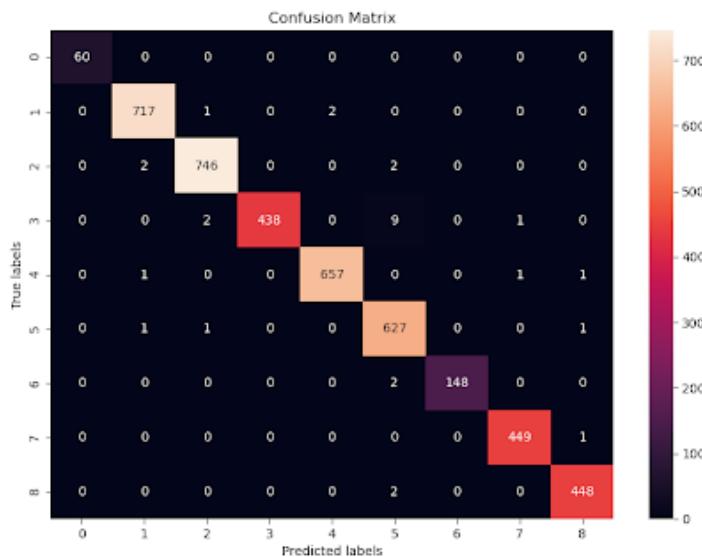
Tabell 4: Sammenligning av presisjon på test-, validering og treningssett

sett	Test	Validering	Trenings
Presisjon	0.9914	1.0000	0.9969
<i>Test- validering og treningssett</i>			

Basert på resultatene presentert i tabellen, ser man at presisjonene til testsettet er noe lavere enn resultatene fra trenings- og valideringssettet, dette er forventet. Likevel er den faktiske presisjonen svært tilfredsstillende, med korrekt prediksjon i 99.14 prosent av tilfellene. Dette viser at modellen ikke overfitter, som kan være et problem når man holder på med trening av store mengder data.

#### 4.1.3 Forvirringsmatrise

Et nyttig verktøy for å vise styrker og svakheter til en modell, er å se på modellen sin forvirringsmatrise. Her kan man se i hvilke tilfeller modellen har gjort feilaktige prediksjoner.



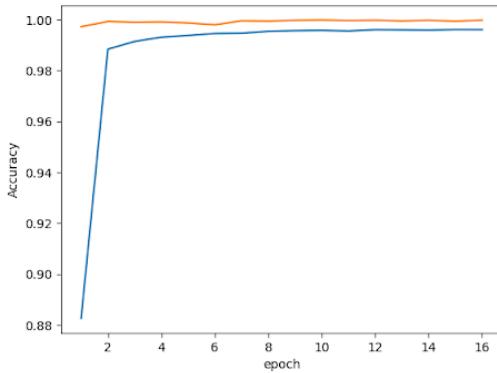
Figur 12: Forvirringsmatrise. Antall feiltaktige prediksjoner  
0-8: 20, 30, 50, 60, 70, 80, oppheving 80, 100, 120

Fra figuren ser man at det kun er et tilfelle hvor modellen har predikert feil mer enn to ganger. Det spesifikke tilfellet er hvor modellen har predikert 80km/h, mens det faktiske skiltet har vist 60km/h, denne feilen er gjort 9 ganger. Siden det er svært få feiltagelser, er det vanskelig å si hvor mye man skal legge i denne dataen, det er derimot en ikke triviell økning fra resten og derfor verdig å nevne.

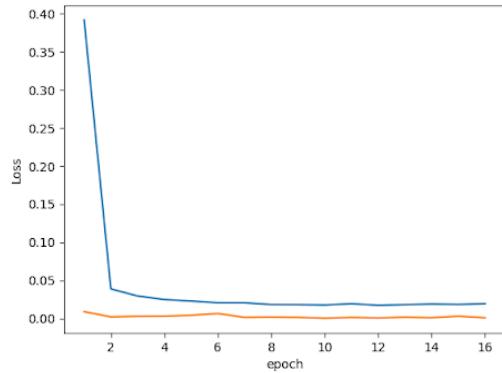
Et annet poeng som også burde representeres, er at noen skilt er overrepresentert sammenlignet med andre. Det er for eksempel mange flere 60km/h, 70km/h og 80km/h skilt sammenlignet med 20km/h og 30km/h. Dette kan også være en av faktorene til at nettopp dette tilfellet hadde noe høyere feil.

#### 4.1.4 Utvidelse av datasettet

Selv om prosjektets målsetning var å kunne predikere i sanntid på fartsgrenseskilt, var det ønskelig å bygge videre på prosjektet i fremtiden. Et av elementene som var interessant å se på, var om det var mulig å utvide modellen til å predikere på hvilke som helst skilt, ikke bare fartsgrenser. Resultatene for det utvidede datasettet blir trent opp på samme måte for det begrensede datasettet, slik at resultatene gir et godt bildet på hvordan modellen håndterer flere typer skilt.



Figur 13: Presisjon trening og validering



Figur 14: Tap trening og validering

De to figurene viser at oppførselen til modellen når den trener på det utvidede datasettet med 43 forskjellige klasser, overordnet er svært lik det begrensede datasettet med kun trafikkskilt. Fortsatt starter valideringssettet med høyere presisjon og lavere loss enn treningssettet, før det stabiliserer seg ved epoke 3-4. I motsetning til det begrensede datasettet, stopper økningen her allerede på epoke 16.

Tabell 5: Resultater utvidet, epoke 1-4

Epoke	Train acc	Valid acc	Train loss	Valid loss
1	0.88267	0.99723	0.39176	0.00908
2	0.98842	0.99930	0.03895	0.00228
3	0.99145	0.99900	0.02961	0.00308
4	0.99313	0.99911	0.02506	0.00320

*Presisjon og tap for trening- og valideringssett epoke 1-4*

Tabell 6: Resultater utvited, epoke 13-16

Epoke	Train acc	Valid acc	Train loss	Valid loss
26	0.99600	0.99950	0.01825	0.00184
27	0.99590	0.99975	0.01908	0.00111
28	0.99612	0.99940	0.01856	0.00321
29	0.99610	0.99981	0.01954	0.00102

*Presisjon og tap for trening- og valideringssett epoke 26-29*

Resultatene fra kjøringen ser vi her er nokså like de fra det begrensede datasettet, men med noe

---

lavere presisjon og høyere loss som generell regel. Likevel er godt over 99 prosent presisjon for både trenings- og validering svært bra, her også er det nødvendig å se på det uavhengige testsettet.

Tabell 7: Sammenligning av presisjon på test-, validering og treningssett

Sett	Test	Validering	Trenings
Presisjon	0.9789	0.9998	0.9961

*Sammenligning av presisjon på test-, validering- og treningssett*

Basert på resultatene presentert i tabell 8, ser man at treningssettet gir et bedre bildet på den faktiske presisjonen til modellen på uavhengig data. Likevel er den faktiske presisjonen tilfredsstillende, med presisjon på 97.89. Selv om dette som forventet er lavere, er det fortsatt et svært brukbart resultat, som tyder på at metodene vi har brukt også er skalerbar til å inneholde langt flere skilt enn kun fartsgrenser.

#### 4.1.5 Sanntidsdeteksjon

Når det gjelder statistikk for hvor bra en cascade classifier er, vil det være tidkrevende å teste i trafikken. Det vi observerte var ofte at andre runde skilt noen ganger ble gjenkjent som et fartsgrenseskilt. Noe annet vi bet oss merke i var at under testing i sterkt motlys eller på natten var det problemer med fokus og hvitbalansen i bildet. Dette motiverte oss til å justere på disse verdiene. Vi implementerte en løsning slik at når en bruker trykker et sted på skjermen skulle den justere fokus og lys ut fra dette området i kameraet. Dette gjorde at skilt ikke ble overeksponert på natten og underekspонert ved sterkt motlys.

Vi kjørte også treningssettet gjennom cascade klassifisereren og fikk statistikk på hvor mange ganger den fant et skilt i bildet og hvor mange ganger den fant ingen eller flere skilt i bildet.

Tabell 8: Resultater fra cascade classifier på treningsdata

Ingen skilt	Et skilt	Flere skilt	Antall bilder
3752	9444	4	13200

*Resultater fra cascade classifier på treningsdata*

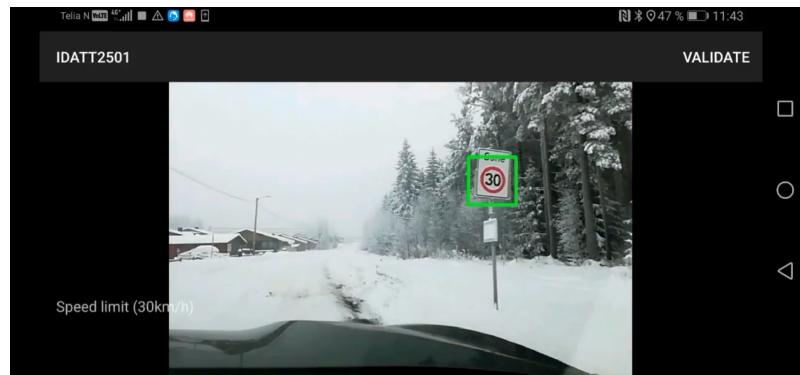
Ut fra resultatene i tabellen kan en se at den finner skilt i rundt tre fjerdedeler av bildene. Det kan tenkes at det er vanskelig å finne objekter i noen av bildene fordi de er veldig utydelige og av lav oppløsning. Bildene vi tar i applikasjonen er derimot veldig mye høyere oppløsning og forventes å gi bedre deteksjonsrate. Vi testet med bilder av god kvalitet og disse hadde den ingen problemer med å finne skiltene. Ettersom applikasjonen derimot skal benyttes på en bil i bevegelse kan dette by på utfordringer knyttet til uskarpe bilder.

#### 4.1.6 Android Applikasjon

Android Applikasjonen er delt opp i to aktiviteter. En kameraaktivitet hvor prediksjonene blir gjort, og en valideringsaktivitet hvor brukeren kan håndtere prediksjonene.

##### Kameraktivitet

Ved åpning av applikasjonen, får man opp en kameravisning, med noen valg.

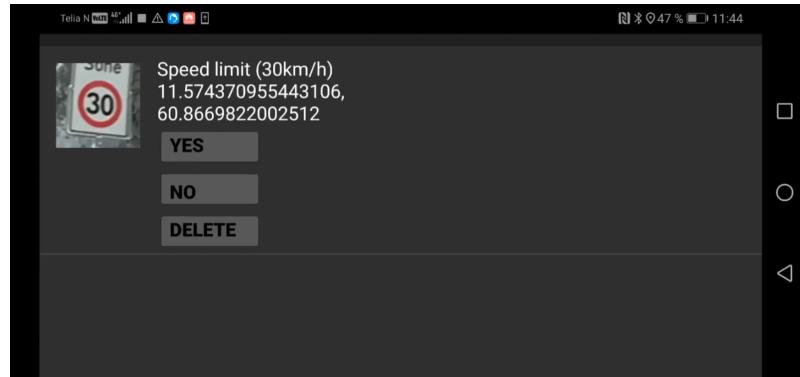


Figur 15: Kameraaktivitet under testing

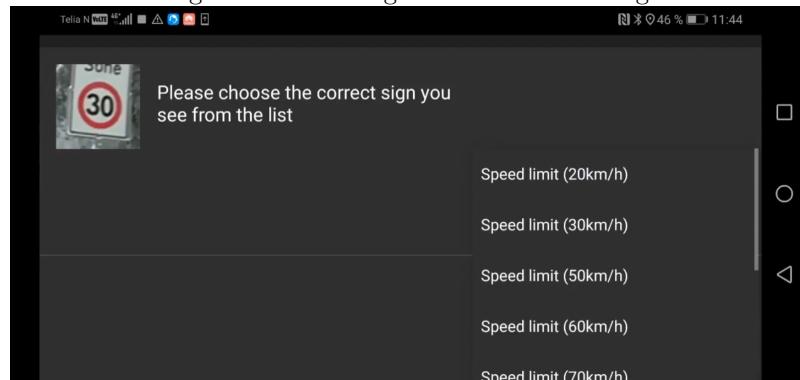
Når applikasjonen detekterer et fartsgrenseskilt, lages en grønn ramme rundt skiltet sin posisjon i bildet. Det som er innenfor rammen blir brukt som input av modellen til prediksjonen. Tilbakemelding på prediksjonen blir gitt i sanntid til brukeren nederst til venstre i bildet. Øverst til høyre i bildet kan man trykke på "Validate", for å bli tatt videre til valideringssiden.

#### Valideringsaktivitet

Valideringssiden gir brukeren mulighet til å gå gjennom prediksjonene som er gjort, samt at den viser geolokasjonen som er registrert på den gitte prediksjonen.



Figur 16: Valideringsaktivitet fremvisning



Figur 17: Valideringsaktivitet oppdatere skilt

Aktiviteten lar brukeren gå over alle prediksjoner som er gjort, og godkjenne dem. Dersom prediksjonen viser seg å være feil, kan brukeren manuelt oppdatere skilttypen på en gitt prediksjon fra en liste over alle skilt. Til slutt så kan også brukeren slette prediksjonen hvis bildet som er predikert ikke er et trafikkskilt. Etter en validering av et skilt vil skiltet og prediksjonen bli fjernet

---

fra listen, så brukeren ikke validerer samme skilt flere ganger. Per nå så er det ikke implementert noen form for lagring av prediksjonene som er gjort, utenom en midlertidig lagring i minne mens applikasjonen er i bruk. Hvordan dette på en hensiktsmessig måte kan implementeres i fremtiden er forbeholdt eventuelle videre utvikling av applikasjonen. Gruppens betraktninger rundt dette blir presentert i diskusjonsdelen av denne rapporten.

#### Fysisk størrelse og ytelse

En av bekymringene før prosjektstart, var at et produkt som dette ikke ville være mulig å kjøre på mobile enheter som følge av begrenset kapasitet på hardware. Applikasjonen viser at dette ikke er tilfellet, men det er fortsatt viktig å ta opp applikasjonen sine begrensninger i denne forstand også. OpenCV modulene er lagret i appikasjonen, som fører til at applikasjonen blir svært stor. Den fysiske størrelsen er 331MB. Den er også krevende på minne og batteri. Etter en kjøretur ble den maksimale minnebruken til applikasjonen registrert på 147MB, mens den gjennomsnittlige minnebruken lå på 11MB.

Applikasjonen har blitt testet på to forskjellige Android enheter. En Samsung Galaxy S9 plus, og en Huawei P20 Lite. Samsung telefonen er relativt kraftig selv om den er et par år gammel. Til sammenligning har Hauwei telefonen begrenset ytelse. Gjennom testing ble det klart at Huawei telefonen sitt svakere hardware gjorde at den ikke var egnet til å detektere trafikkskilt ved høye hastigheter. Samsung telefonen klarte derimot dette uten store problemer, da den kjørte applikasjonen med betraktelig høyere bildefrekvens og kameraoppløsning. Den hadde en gjennomsnittlig framerate på 10-12 bilder i sekundet. Samsung Galaxy S9 plus er som nevnt allerede en eldre telefon, og de fleste moderne medium til high-end Androidtelefoner vil derfor kunne kjøre applikasjonen uten store problemer med ytelse. Vi forventer også at disse ville kunne kjøre med høyere frekvens og klarere bilder. Av den grunn vil det også være mulig å oppdagte enda flere skilt.

## 4.2 Ingeniørfaglige resultater

I denne delen av rapporten vil de vitenskapelige resultatene samt det helhetlige produktet satt opp mot de konkrete målene utarbeidet sammen med produkteier i Visjonsdokumentet.

### 4.2.1 Produktets funksjonelle egenskaper

I Visjonsdokumentet ble det gjort rede for at den første og viktigste egenskapen til applikasjonen var å kunne laste inn en modell, og bruke OpenCV til å detektere og predikere trafikkskilt i sanntid. Samtidig var det et ønske om å legge til rette for å bekrefte og forkaste resultater, slik at brukere kan validere data som etter hvert kunne brukes i kartdata.

Applikasjonen bruker Pytorch mobile for å lese inn maskinlæringsmodellen. En Haar Cascade.xml fil brukes ved hjelp av OpenCV sin CascadeClassifier, som lar applikasjonen detektere bilskilt basert på haarcascade filen sitt innhold. Når applikasjonen detektere et fartsgrenseskilt, blir den predikerte fartgrensen gitt som feedback til brukeren og lagt til validering. Fra denne aktiviteten kan brukeren vurdere om prediksjonen er riktig. Hvis ikke vil brukeren kunne velge blant listen med fartsgrenser, eller forkaste prediksjonen dersom det ikke er et gyldig trafikkskilt. Validering av prediksjons aktiviteten har foreløpig ingen funksjon utover å visualisere og skape en interaksjon med resultatet for brukeren, men legger til rette for fremtidig utvikling ved ønske om å lagre informasjonen om prediksjonene. Forslag av mindre relevanse slik som kartvisning og brukerinnlogging ble prioritert bort. Likevel er både geolokasjon og administrering inkludert, da det i løpet av prosjektarbeidet har vist seg som hensiktsmessig. Både med tanke på det ferdige produktet, men også for videre arbeid og utvidelser. Mulighet til å predikere på andre trafikkskilt enn kun fartsgrenseskilt ble sett på som en ekstra utfordring dersom gruppen kom i mål med den initiativene oppgaven. Gruppen har derfor også utviklet en modell som baserer seg på samme arkitektur som den opprinnelige modellen, men som er trent opp på flere typer trafikkskilt.

Ettersom Haar Cascade er en objekt deteksjon modell vil den detektere objekter i hvert bilde

---

som tas opp av enhetens kamera. Dette resulterer i at samme skilt kan bli predikert gjentatte ganger. Dette vil man kunne se i valideringsaktiviteten da samme skilt dukker opp flere ganger etter hverandre i listen over prediksjoner. Disse vil derimot ha veldig lik lokasjon og i en videreutvikling kan denne, sammen posisjon i kamerasynsvinkelen, brukes til å klassifisere to bilder som samme skilt

#### **4.2.2 Ikke funksjonelle egenskaper og andre krav**

Prosjektet har innfridd på de ikke funksjonelle kravene definert i visjonsdokumentet. Det har resultert i en Android Applikasjon som er tilgjengelig på alle mobile enheter med kamera som kjører Android 5.0 eller nyere. Den er utviklet i native android, og bruker OpenCV for tolkning av trafikkskilt.

All koden i prosjektet er dokumentert, slik at hvem som helst skal kunne sette seg inn i kildekoden til prosjektet. Dette gjelder spesielt selve Android applikasjonen, men også maskinlæringskoden i sitt eget repository slik at begge disse delene kan gjenbrukes og videreutvikles.

Tilknytning opp mot OpenStreetMap ble nevnt som en mulig fremtidig utvidelse sammen med produkteinne på første møte, men det var likevel et ønske om enkel behandling av dataen som applikasjonen innhenter. Valideringssiden legger til rette for fremtidig utvidelse til kart-api, som kan bruke trafikkskiltene sine lagrede geolokasjoner.

Det endelige produktet følger WCAG sine prinsipper om begripelighet. Prosjektets har sammen med produkteinne blitt definert som først og fremst et forskningsprosjekt fremfor et systemutviklingsprosjekt. Selv om applikasjonen kan sees på som et verktøy for å demonstrere forskningen som er gjort, har det fortsatt vært fokus på at den skal være enkel å bruke for hvem som helst som ønsker å benytte seg av teknologien. Av den grunn har applikasjonen store leselige knapper og høye kontrastnivå der nødvendig.

### **4.3 Administrative resultater**

Denne delen av rapporten tar en gjennomgang av hvordan prosjektarbeidet har blitt gjennomført. Dette vil bli knyttet opp mot eventuelle utfordringer, slik at det kan være av nytte for de som ønsker å gjennomføre tilsvarende prosjekter.

#### **4.3.1 Fremdriftsplan**

Som nevnt i delkapittel 3.2.1 har prosessen blitt delt opp i fem ulike deler. Å fastsette datoer var noe utfordrende, og ikke alltid hensiktsmessig. Den endelige datoene for innlevering, samt et tidsperspektiv for visning av prototypen har vært de to store milepælene.

Gruppen hadde som mål om å jobbe en 8 timers arbeidsdag en gang hver uke i starten av semesteret, for så å øke dette til to arbeidsdager etter hvert som man nærmet seg fristen. Gruppen var også klar over at det enkelte perioder ville bli nødvendig å prioritere andre fag, med tanke på eksamener og semesterprøven som krever ekstra kapasitet. Dette skulle kompenseres med ekstra arbeidsdager de periodene hvor gruppemedlemmene hadde ekstra arbeidskapasitet. Spesielt etter 19. desember, som var siste aktuelle eksamensdato for gruppemedlemmer dette semesteret.

Dette målet ble oppnådd på en relativt tilfredstillende måte, det var derimot noen forventede avvik fra planen. Som følge av diverse innleveringer og prøver, ble slutten av oktober noe travel før framvisning av MVP for produkteinne. Det ble avtalt at prototypen skulle vises en gang i slutten av oktober. Prototypen ble ferdigstilt 27 oktober, som var noe etter gruppens relative målsetning, men likevel innenfor fristen. Den endelige fristen for innlevering var 3. januar. I utgangspunktet ønsker gruppen å bli ferdig før jul, men anerkjente at det kunne bli nødvendig å jobbe også i romjulen for å få nok timer. Som følge av en spesielt hektisk November hvor det kun ble jobbet 1.,

---

2. og 30. med fordypningsprosjekt, trengte gruppen å bruke noen dager i romjulen på å ferdigstille rapporten. Utenom dette gikk fremdriften i prosjektet som beskrevet i planen.

#### 4.3.2 Tidsforbruk

Hovedregelen for dette prosjektet har vært at gruppen jobber sammen i den grad det lar seg gjøre, i arbeidsdager som går over 8 timer, med noen halve dager etter behov. Selv om det naturligvis blir noe arbeid alene avhengig av kapasitet, utgjør dette såpass lite at det ikke er behov for å se på individuelle timer. Et estimat på timer jobbet selvstendig er 10 timer per person, og til sammen er det blitt jobbet 160 timer sammen i felleskap. Dette utgjør et totalt antall timer per person på 170 timer. Den vedlagte timeoversikten (Figur 18) gir et bilde på hvor intense periodeene har vært, og når gruppene har lagt inn mesteparten av arbeidsinnsatsen. En periode utgjør en halv måned, eller omtrent to uker. Fra rapporten ser man en gradvis stigning opp mot de to milepælene, prototypen og innleveringen. I disse periodene ble det henholdsvis 28 og 40 fellestimer.

Det er vanskelig å gi et klart skille på hvor mye av tiden som er fordelt på research og systemutvikling. For det første er oppgaven forskningsbasert, slik at en stor del av tiden naturligvis går til å sette seg inn i gitte teknologier. Selve programmeringsdelen handler ofte om å prøve, så feile, deretter tilegne seg ny kunnskap for å se hva som ikke er rett. Av denne grunn har det ikke vært mulig å beregne hvor mye av tiden som har gått til forskning og systemutvikling spesifikt. Det kan derimot være verdt å nevne at fordelingen mellom tid på applikasjon og modell har vært tilnærmet 50/50. I starten ble det litt overvekt på applikasjon for å teknologier til å fungere, mot slutten ble det ekstra fokus på å optimalisere presisjon på modellen.

## 5 Diskusjon

For å avgjøre hvorvidt prosjektarbeidet har blitt løst på en tilfredsstillende måte, er det nødvendig å reflektere over resultatene og betraktingene som er presentert i denne rapporten. For et prosjekt som bærer forskningspreg er det naturligvis ikke alltid like enkelt å definere håndfaste parametere som avgjør om prosjektet gir gode svar på problemstillingen eller ikke. Det er mulig å ikke løse den konkrete problemstillingen, men likevel å ha gjort forskning av verdi og nytte for videre arbeid. Basert på resultatene som har blitt presentert i tidligere kapitler i denne rapporten, er gruppen sin oppfatning at selve produktet er tilfredsstillende ovenfor rammene satt i Visjonsdokumentet sammen med produkteinier. Det er likevel strengt nødvendig å reflektere over resultatene man har oppnådd, og komme med forslag til videre utvikling og nåværende svakheter. Den røde tråden gjennom prosjektet var aldri *Lag en mobilapplikasjon for sanntidsdeteksjon* men heller *I hvilken grad er det mulig eller hensiktsmessig å lage en mobilapplikasjon for sanntidsdeteksjon*. Dette kapittelet vil være en omfattende diskusjon rundt gruppens resultater i prosjektets områder, som til slutt skal gi et helhetlig bilde som gir et svar på denne problemstillingen.

### 5.1 Datasett

Datasettet som den nåværende modellen bruker, klassifiserer basert på ni forskjellige typer fartsgrenseskilt. For dette spesifikke prosjektet sitt omfang, så er det tilstrekkelig for å gi et bildet på om sanntidsdeteksjonen er gjennomførbar. Ved å ha ni forskjellige klasser er det mulig å få både et godt bilde på både hvordan modellen sin overordnede presisjon blir, samt hvordan applikasjonen fungerer i praksis. I første omgang handler et slikt type prosjekt om det i det hele tatt er mulig å klassifisere på denne måten, første prioritet for et datasett blir derfor å ha nok data til å kunne predikere med høy presisjon, uten at modellen overfitter. Det første datasettet som ble forsøkt brukt, inneholdt totalt 600 bilder for trening, test og validering, dette fant gruppen fort ut at ikke ble tilstrekkelig.

Datasettet som ble tatt i bruk i stedet var betraktelig større, med totalt 17 520. I utgangspunktet høres dette ut som et ideelt datagrunnlag, men det er likevel noen begrensende faktorer som er

---

viktig å ta til betraktnng. For det første så viste det seg nok en gang at dette ikke er tilstrekkelig for å trenre opp en sikker modell, settet ble derfor utvidet og utjevnet med bilde augmentasjon. Slik fikk vi 20 000 bilder per skilt. For det andre så er skiltene fra datasettet tatt fra tyske veier. Disse er sammenlignbare med de norske, men har blant annet en hvit sirkel rundt skiltene som de norske skiltene ikke har. Basert på atferden til applikasjonen, hvor skiltene veldig ofte predikerer riktig, er det rimelig å anta at dette fortsatt er tilfredsstillende for bruk på norske veier. Likevel vil en slik forskjell bidra til at man ikke kan stole hundre prosent på presisjonen som man får når man predikerer på det uavhengige testsettet hvor også alle skiltene vil være på det tyske formatet.

En annen tydelig begrensning på datasettet, er at det ikke inkluderer 40 km/h eller 90 km/h. Som nevnt tidligere er dette en begrensning man kan leve med i denne fasen av et prosjekt, men som derimot blir helt nødvendig å inkludere om man ønsker å videreutvikle prosjektet slik at det kan brukes til det tiltenkte formålet. Modellens høye presisjon tyder likevel på den uten problemer vil håndtere disse skiltene også, gitt at den får tilstrekkelig med data. Dette tar oss videre til et annet interessant punkt som kanskje ikke er like åpenbart hva som er mest riktig, dette er hvorvidt det er hensiktsmessig å utvide modellen til å inkludere trafikkskilt utover kun fartsgrenseskilt. Resultatene til dette prosjektet har vist at også dette kan gi svært gode resultater på presisjon, selv med godt over 43 forskjellige klasser. I dette prosjektets rammer er det spesifikt definert fartsgrenser som den klare førsteprioritet. Nå som dette er vist mulig, sammen med det utvidede datasettet, er det interessant å se om det er mulig å klassifisere på samtlige norske skilt.. Her er det noen ytterlige utfordringer som dukker opp som ikke har vært like relevante mens datagrunnlaget har vært begrenset. For det første vil lokale forskjeller på skilt spille en større rolle. Et tyskt 60km/h skilt vil være tilnærmet likt 60km/h på norske skilt. Man har derimot norske trafikkskilt som "Elfare" som ikke eksisterer i det tyske skiltregisteret og som konsekvens av dette ikke eksisterer i datasettet som blir benyttet. For å få et komplett datagrunnlag for alle norske skilt, er man derfor avhengig av en god del manuelt arbeid for å innhente nok data til å gi tilfredsstillende nøyaktighet ved prediksjon. En annen utfordring blir i det man skal detektere skilt som ikke er runde, for eksempel vikepliktskilt. For å oppnå dette er man avhengig av flere typer Cascaadefiler, dersom man ønsker å bruke samme løsning som den som er presentert i denne rapporten. Her kan en annen løsning slik som benytelse av YOLO være mer hensiktsmessig, da denne støtter deteksjon av flere klasser samtidig.

## 5.2 Maskinlæringsmodell

Som presentert i resultatdelen, har maskinlæringsmodellen gått gjennom mange forskjellige iterasjoner og utvidelser. Den kanskje viktigste endringen for resultatene til modellen ble gjort på selve datasettet. Det ble klart at det var nødvendig å gi et tilnærmet likt distribuert datasett for at modellen skulle trenre riktig. STN viste seg å være en annen kritisk forbedring, som gjorde at modellen fikk mer standardiserte bilder og trenre på. Som konsekvens av dette eliminerte man mye av bakgrunnen, slik at selve skiltet tok opp en større del av bildet. Spesielt viktig var dette etter transformasjonene som ble gjort for å utvide datasettet, da posisjonen til selve skiltet i bildet ble mer forutsigbart.

Disse utvidelsene bidra i stor grad til den høye presisjonen til modellene, samt at bildene som ble registrert i Android Applikasjonen var kompatible med modellen som ble laget. Godt over 99 prosent presisjon er meget tilfredsstillende resultat. Mye av jobben har gått til å gjøre input bildene fra applikasjonen og bildene modellen trener på, så like som mulig. Dersom man skal få ut så mye av modellen potensial som mulig, er dette et nødvendig arbeid. Selv om gruppen har kommet langt med dette arbeidet, er det her rom for forbedring og videre utforskning.

Videre var det viktig for gruppen å teste om modellen var skalerbar på og kunne benyttes på flere trafikkskilt enn kun fartsgrenseskilt. Dette viste seg å være tilfellet. Med en presisjon på 97.89 prosent, er utgangspunktet for videre jobb med mange forskjellige typer trafikkskilt svært godt. Som nevnt tidligere i dette kapittelet er det en jobb å gjøre på selve datagrunnlaget dersom alle norske skilt skal med, men forsökene som er presentert i rapporten tyder på at modellen er egnet for denne utvidelsen.

---

### 5.3 Android Applikasjon

Applikasjonen som ble utviklet er relativt enkel. Det ble nevnt i Visjonsdokumentet at applikasjonen ikke trengte å være fryktelig komplisert, men hadde et par funksjoner som måtte være på plass for å være tilfredsstillende. Dette var selve kameraaktiviteten, samt en enkel håndteringsaktivitet som kunne la brukeren håndtere prediksjonene som ble gjort. Denne aktiviteten utgjør sitt formål på dette stadiet av prosjektet, men det er klart at det er eventuelle forbedringer som burde gjøres dersom man skal ta prosjektet et steg videre.

Geolokasjon registreres for alle prediksjoner som blir gjort. Per nå så blir ikke denne informasjonen brukt til mer enn en enkel valideringsaktivitet for brukeren, informasjonen blir ikke lagret noe sted da dette var utenfor omfanget til oppgaven. Det er derimot ikke mye som skal til for at informasjonen skal kunne benyttes på en hensiktsmessig måte. For å gjøre dette er det et par ting som en må ta til betraktning. For det første registerer applikasjonen alle prediksjoner den gjør. Dette betyr at dersom et fartsgrenseskilt blir fanget av kameraet i applikasjonen over en hvil periode, vil det registreres flere prediksjoner for det gitte skiltet. Avhengig av hvordan en eventuell utvidelse er planlagt gjennomført, kan dette være enten positivt eller negativt. Det første som er viktig å betrakte, er at ikke alle bildene vil være av god kvalitet selv om de detekteres av applikasjonen. Dette betyr at det er en sjanse for at den første prediksjonen blir feil som følge av at bildet av skiltet er u tydelig, mens den neste prediksjoner, samt alle som kommer etter blir riktige. Ved å registrere alle prediksjonene, kan man la det være opp til brukeren å slette eventuelle u gyldige prediksjonene slik det er lagt opp til i applikasjonen nå. Et annet alternativ er å kun registrere én prediksjon per lokasjon. Som følge av at applikasjonen bruker den mobile enhetens lokasjon, vil lokasjonen applikasjonen håndterer oppdatere seg noe sporadisk. På denne måten kan man anta at flere prediksjoner av samme skilt også vil registrere samme lokasjon. Dette betyr at man kan bruke skiltenes lokasjon til å bekrefte eller avkrefte om man har samme skilt, med relativt høy presisjon. Denne antagelsen må naturligvis utforskes og testes før man kan si med sikkerhet at det er en løsning som fungerer i praksis. En begrensning dette gir, er dersom man ønsker å inkludere mer enn kun fartsgrenseskilt. Her vil det naturligvis være et behov for å kunne registrere flere prediksjoner per geolokasjon.

### 5.4 Sanntidsprediksjon

Haar Cascade filen som ble brukt i prosjektet ble som nevnt tidligere ikke trent selv opp av gruppen, men heller funnet på nett av tidsmessige grunner. Det ble gjort et forsøk på å trenere opp en egen Haar Cascade model selv med 445 positive bilder og 5363 antall negative bilder som treningsdata. Men dette var ikke nok treningsdata da man gjerne skal ha minst over 2000 av både de positive og negative bildene blant treningsdataen. Det resulterte i at modellen ga svært mange falske positive deteksjoner. Langt flere enn den ferdigtrente som senere ble tatt i bruk. Så med grunnlag i manglende treningsdata valgte gruppen å benytte seg av en forhåndstrent Haar Cascade fil. Denne modellen var heller ikke optimal, men det beste alternativen av de to.

At modellen ikke var optimal resulterte i at deteksjonen av skiltene ikke alltid ble helt helt korrekte, da applikasjonen i noen tilfeller kan detektere noe som et skilt selv om det ikke er det. Men ettersom applikasjonen har valideringsaktivitet brukeren kan benytte er det ikke et massivt problem, og utgjør ingen stor negativ effekt på brukbarheten til applikasjonen. Hadde situasjonen vært at den ikke detekterte noen skilt hadde vært et større problem, da man hadde gått glipp av veginformasjon. I tilfelle hvor deteksjonen ikke presterer optimalt kan det tenkes at det er bedre å ta inn litt for mye data og la brukeren bestemme hva som er riktig eller galt, enn å ta inn for lite og miste prediksjon av skilt.

Basert på denne informasjonen, samt svakhetene Haar Cascade filen kun kan klassifisere runde skilt for oss, er et naturlig spørsmål om det ikke ville vært mer hensiktsmessig å bruke for eksempel Yolo. Å kunne detektere skilt av flere forskjellige former er et område hvor applikasjonen har et stort forbedringspotensial, men som gruppen mistenker vil kreve en god del prøv og feil. For det første er YOLO svært mye mer krevende å kjøre enn Haar Cascade. Applikasjonen krever

---

allerede en god del fra den mobile enheten ved kjøring, og mistanken er at en løsning som YOLO ikke vil kunne gi tilbakemeldinger i sanntid slik som Haar Cascade. Til sammenligning er å bruke flere Haar Cascade filer er kanskje ikke den mest skalerbare løsningen, og det er vanskelig å si hvor komplisert dette vil være å implementere. Uavhengig av hva som viser seg å gi best resultater, er det ingen tvil om at dette vil være svært nyttig forskning for problemstillingen, og burde være av høy prioritet dersom dette prosjektet jobbes videre på.

## 5.5 Samarbeid mellom teknologier

En av de største utfordringene med å jobbe med et prosjekt av en slik art, er at man aldri helt kan se for seg hvilke problemer en vil støte på. Spesielt i tilfeller hvor man ikke har jobbet med de aktuelle teknologiene ved tidligere anledning. Selv om disse tilfellene ofte er spesifikke for hvert prosjekt, er det fortsatt relevant å ta det opp i denne delen av rapporten da erfaringene kan brukes på generell basis. Først og fremst så viste deg seg å være svært utfordrende å få OpenCV til å fungere i Android Studio. Som nevnt i kapittel 3.2.3.1 gjorde en bug i Android Studio det umulig å importere OpenCV modulene inn i nyere prosjekter på tiltenkt måte. Selv om løsningen kan virke enkel, er det ikke nødvendigvis like lett å komme frem til gode løsninger på problemer man ikke helt forstår selv. For det første krever det en del kunnskap om de aktuelle teknologiene for å i det hele tatt vite hva det er man leter etter. Å tilegne seg slik kunnskap er naturligvis et nødvendig steg i ethvert prosjektarbeid, men gjør at relativt raske løsninger ikke blir like rett frem likevel, derfor er det viktig å sette lys på dem. Gruppen ønsket å finne en mer generell løsning på dette spesifikke problemet, men etter to hele arbeidsdager hvor man lette etter gode skalerbare løsninger, var det ikke lenger forsvarlig å bruke tid på dette. Noe av de samme utfordringene dukket opp ved lasting av Pytorchmodellene i Android Studio, dog i mindre grad. Dette er utfordringer som er vanskelig å vite hvor lang tid tar før man setter i gang, og derfor viktig å ha i bakhodet når man planlegger tid man skal bruke til prosjektarbeidet.

## 5.6 Prosjektarbeidet

Gruppen sier seg fornøyd med gjennomføringen av prosjektet, til tross for at det har vært utfordrende å kombinere med tre andre fag i et svært hektisk høstsemester. Arbeidskapasiteten har naturligvis vært begrenset i enkelte perioder, som har ført til uker hvor prosjektet har blitt nedprioritert. Den store ulempen med dette er at det i et slikt prosjekt er vanskelig å legge fra seg for så å plukke opp et par uker senere, spesielt i startfasen. Dette kommer som følge av alle de nye teknologiene som en må sette seg inn, som det kanskje hadde vært heniktsmessig å jobbet med kontinuerlig over en lengre periode. I ettertid ville gruppen prøvd å satt av en to til tre dager på rad i oppstartfasen, slik at starten av dagen ikke ble like preget av at en igjen måtte sette seg inn i hva man gjorde for en uke siden. Det skal likevel sies at gruppen er fornøyd med å ha jobbet kontinuerlig å målrettet gjennom hele semesteret, og at man må regne med litt ekstra intensitet før milepæler.

En av grunnene til at prosjektarbeidet kom i mål, opplever gruppen kommer av et stort fokus på å prøve ut og sette seg inn i mange forskjellige teknologier. I rapporten kommer det frem flere eksempler på teknologier som ikke har blitt brukt i prosjektet, men som likevel har blitt undersøkt og forsøkt implementert. Selv om det kan føles umotiverende å bruke flere timer på en teknologi man ikke ender opp med å bruke, er det strengt nødvendig dersom man skal ende opp med et godt resultat. Gruppen har jobbet med modellen og applikasjonen parallelt, slik at valget av teknologier i den ene delen, har blitt gjort i lys av behovet på den andre. Som en konsekvens av dette har det blitt gjennomgått et stort utvalg alternativer. Noen av de som ikke ble valgt har likevel gitt verdifull læring og innsikt i hvilke styrker og begrensninger vår løsning har, og hvilke mulige forbedringer som potensielt kan bli gjort.

Det er blitt gjennomført møter med veileder jevnt over prosjektabeidet. Dette var ukentlig i starten av prosjektet, og ble mindre aktuelt etter hvert som prosjektet tok form. Hovedfokuset har alltid vært på progresjonen i prosjektet, prioriteringer, samt en gjennomgang av utfordringer og betraktninger. På denne måten har både gruppen og veileder det klart for seg hvordan status

---

er på prosjektarbeidet, og hvordan det jobbes for å komme seg videre. Dette har vært nyttig av flere grunner, først og fremst fordi tilbakemeldingene har hjulpet gruppen til å holde fokus på målsetninger og arbeidsoppgaver i prosjektet. Samtidig mener gruppen at det er viktig at begge ledd i prosessen har en oversikt over jobben som blir gjort, og hvilke utfordringer en møter på underveis.

## 6 Konklusjon

Prosjektet har resultert i en Android Applikasjon som ikke bare kan detekterer trafikkskilt, men også klassifiserer disse i sanntid. Modellen som er laget for applikasjonen predikerer riktig 99.14 prosent av gangene med ni forskjellige klasser fartsgrenseskilt, men også 97.89 etter datagrunnlaget ble utvidet til 43 ulike typer trafikkskilt. Dette er lovende for videre arbeid, og viser at det konvolusjonelle nevrale nettverket som brukes for å trenne modellen også er skalerbar for mange forskjellige typer skilt. Sanntidsdeteksjon viste seg å være en stor utfordring. Haar Cascade klassifisering er overlegen på enkelte områder, men mangefull på andre. Den er svært rask, og fungerer ypperlig for sanntidsdeteksjon av en klasse skilt, i vårt tilfelle runde. Det er derimot svært vanskelig og ressurskrevende å trenne opp en selv, og det er ikke nødvendigvis lett å finne en som fungerer tilstrekkelig til sitt eget tilfellet. Om en ønsker å utvide til flere skilt av forskjellige fasonger, er det fortsatt ikke åpenbart for gruppen om det beste alternativet er Haar Cascade, eller om det er mulig å implementere en mer ressurskrevende teknologi som YOLOv3 på en hensiktsmessig måte.

Spørsmålet denne rapporten skulle gi svar på var i hvor stor grad det er gjennomførbart og hensiktsmessig å lage mobile applikasjoner som benytter seg av klassifikasjon i sanntid. Prosjektarbeidet har vist at det er et stort potensiale for bruk av maskinlæring med mobile enheter, likevel er det viktig å være forberedt på de utfordringene man ikke kommer foruten. Det er først og fremst en utfordring å utvikle noe nytt hvor så mange teknologier skal spille på lag, spesielt dersom det er teknologi man ikke har jobbet med tidligere. Derfor er det viktig å gå ut bredt, og vurdere og prøve ut et stort utvalg teknologier før man tar endelige avgjørelser. Når det er sagt så er mulighetene uendelige, og med den raske utviklingen av mobile enheter vil maskinlæring snart være et tilgjengelig verktøy hvem som helst med en mobiltelefon.

---

## Referanser

- [1] Import module from source: Next/finish buttons disabled. URL: <https://issuetracker.google.com/issues/195336461>.
- [2] IBM. What is computer vision. URL: <https://www.ibm.com/se-en/topics/computer-vision>.
- [3] Mariana Eugenia Ilas and Constantin Ilas. Towards real-time and real-life image classification and detection using cnn: a review of practical applications requirements, algorithms, hardware and current trends. In *2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pages 225–233, 2020. doi:10.1109/SIITME50350.2020.9292253.
- [4] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. URL: <https://arxiv.org/abs/1506.02025?context=cs>.
- [5] Ali Farhadi Joseph Redmon. Yolov3: An incremental improvement. URL: <https://arxiv.org/abs/1804.02767>.
- [6] Muhammed Nur Talha Kilic. Adaptive touch to histograms— clahe. URL: <https://medium.com/@mntalha.kilic/adaptive-touch-to-histograms-clahe-4b0db004e2bd>.
- [7] Adrien Lescourt. opencv native androidstudio. URL: [https://github.com/leadrien/opencv-native\\_androidstudio](https://github.com/leadrien/opencv-native_androidstudio).
- [8] Mykola. Gtsrb - german traffic sign recognition benchmark. URL: <https://www.kaggle.com/meowmeowmeowmeow/gtsrb-german-traffic-sign>.
- [9] Joseph Nelson. What is onnx? URL: <https://blog.roboflow.com/what-is-onnx/>.
- [10] OpenCV. Cascade classifier. URL: [https://docs.opencv.org/3.4/db/d28/tutorial\\_cascade\\_classifier.html](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html).
- [11] OpenCV. Opencv java documentation (4.5.5-dev). URL: [https://docs.opencv.org/4.x/javadoc\\_index.html](https://docs.opencv.org/4.x/javadoc_index.html).
- [12] protonon. URL: <https://github.com/protonon/tsaraisa/blob/master/haarCascade.xml>.
- [13] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.
- [14] Great Learning Team. What is machine learning? how machine learning works and future of it? URL: <https://www.mygreatlearning.com/blog/what-is-machine-learning/>.
- [15] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001. doi:10.1109/CVPR.2001.990517.
- [16] Chi-Feng Wang. What's the difference between haar-feature classifiers and convolutional neural networks? URL: <https://towardsdatascience.com/whats-the-difference-between-haar-feature-classifiers-and-convolutional-neural-networks-ce6828343aeb>.

---

**Vedlegg**

**Vision Document**

---

**Team 14**

**Prosjektnavn  
Visjonsdokument**

**Versjon 1.1**

---

## Vedlegg

### Vision Document

#### Revisjonshistorie

Dato	Versjon	Beskrivelse	Forfatter
14.09.2021	1.0	Første utkast	Simon Jensen Lars Brodin Østby Stian Mogen
15.09.2021	1.1	Godkjent av veileder	Simon Jensen Lars Brodin Østby Stian Mogen Alexander Holt

---

# Vedlegg

## Vision Document

### Innholdsfortegnelse

1.	Innledning	4
1.1	Referanser	4
2.	Sammendrag problem og produkt	4
2.1	Problem Sammendrag	4
2.2	Produktsammendrag	4
3.	Overordnet beskrivelse av interessenter og brukere	4
3.1	Oppsummering interessenter	4
3.2	Oppsummering brukere	4
3.3	Brukermiljøet	4
3.4	Sammendrag av brukernes behov	5
3.5	Alternativer til vårt produkt	5
4.	Produktoversikt	5
4.1	Produktets rolle i brukermiljøet	5
4.2	Forutsetninger og avhengigheter	5
5.	Produktets funksjonelle egenskaper	5
6.	Ikke-funksjonelle egenskaper og andre krav	5

---

# Vedlegg

## Vision Document

### 1. Innledning

Dette dokumentet beskriver de overordnede kravene og målene for prosjektet i IDATT2505 Fordypningsprosjekt for Team 14. Prosjektet går ut på å utvikle en android applikasjon til en mobil enhet som ved hjelp av maskinlæring i OpenCV skal kunne kontinuerlig filme, lese inn og tolke informasjon om veiskilt, i første rekke med hovedfokus på fartsgrenser. Prosjektet utføres av 3.år Bachelor ingenørarfag, data i høstsemesteret 2021.

### 2. Sammendrag problem og produkt

#### 2.1 Problem Sammendrag

Innsamling av data er essensielt for nytteverdien til en kartjeneste. OpenStreetMap er en open source kartjeneste, som er svært populær blant utviklere og brukere verden over. Vi ser i dag at slike tjenester har et behov for enkel og brukervennlig innsamling av data, som igjen kan brukes til å forbedre og videreutvikle karttjenester som OpenStreetMap. Der enkelte bilprodusenter innhenter sin egen data for bruk i sine tjenester, er det et ønske om et system som kan benyttes av uavhengige brukere, som kan benyttes av fellesskapet i egne prosjekter.

Problem med	Oversikt av fartsgrenser på åpne kart-api som OpenStreetMap
berører	Utviklere som ønsker et kart-api med åpen kildekode
som resultatet av dette	Få alternativer for utviklere som ønsker å utvikle systemer som bruker kart, med mulighet for informasjon om trafikkskilt
en vellykket løsning vil	Med maskinlæring kunne dokumentere trafikkskilt, med innhentet data fra et enkelt og brukervennlig brukergrensesnitt.

#### 2.2 Produktsammendrag

For	Trafikanter
som	Ønsker å kunne tolke trafikkskilt langs veien
“Produktnavn”	Er en mobilapp som kan tolke trafikkskilt i realtime på mobil
som	Trener data til bruk i OpenStreetMap
I motsetning til	Google Maps og andre in house kartløsninger til bilprodusenter
Har vårt produkt	Mulighet for at brukerne selv kan legge til og oppdatere data i en open source karttjenesten

### 3. Overordnet beskrivelse av interessenter og brukere

#### 3.1 Oppsummering interessenter

Navn	Utdypende beskrivelse	Rolle under utviklingen
Produkteier Alexander Holt	Denne personen representerer kunden av produktet. 3D Motion Technologies	Denne personen vil komme med krav og sentrale innspill til produktet under planlegging og utvikling

---

# Vedlegg

## Vision Document

Team	Utviklere av systemet	Disse har som hovedoppgave å utvikle et produkt som tilfredsstiller produkteiers krav og forventninger
------	-----------------------	--

### 3.2 Oppsummering brukere

Navn	Utdypende beskrivelse	Rolle under utviklingen	Representert av
Testbrukere	Disse brukerne er potensielle målgrupper til det ferdige produktet	Disse vil ha som oppgave å gjennomføre testing av produktet og komme med tilbakemeldinger	En gruppe testere
Trafikanter	Brukere som benytter systemet i trafikken for å holde styr på blant annet fartsgrense	Den største brukergruppen som vil være i fokus under utforming og utvikling av produktet	
Systemutviklere	Brukere som benytter seg av innhentet data i egne prosjekter med Open Street Map		

### 3.3 Brukermiljøet

Android mobil: Denne applikasjonen skal kunne kjøre på alle mobile enheter som bruker android 5.0 operativsystemet eller nyere.

OpenStreetMap: Applikasjonen skal kunne fungere med OpenStreetMap ettersom hensikten med applikasjonen er å samle og tolke veiskilt informasjon for plattformen.

### 3.4 Sammendrag av brukernes behov

Behov	Prioritet	Påvirker	Dagens løsning	Foreslått løsning
Brukerregistrering	Lav	Innlogging		Brukeren registrerer selv en bruker ved første innlogging
Administrere datainnsamling	Lav	Bruk		Brukeren kan velge hva som sendes inn av innsamlet data
Tolkning av fartsgrense	Kritisk	Maskinlærings-modell		Brukeren får opp en visning av skiltet når det er gjenkjent av applikasjonen
Tolkning av andre trafikkskilt	Middels	Maskinlærings-modell		Brukeren får opp en visning av skiltet når det er gjenkjent av applikasjonen
Fotobokser				

---

# Vedlegg

## Vision Document

Sosiale medier	Lav	Innlogging		Brukeren skal kunne logge inn gjennom twitter, facebook, google konto
Tilbakemeldinger	Lav	Brukert		Brukert skal gjennom en funksjon i appen kunne kontakte oss for spørsmål og tilbakemeldinger
Kartvisning	Lav	Applikasjon		Brukert skal kunne få opp en visning av hvor vedkommende befinner seg

### 3.5 Alternativer til vårt produkt

Å bruke maskinlæring til å tolke skilt er en populær treningsinnenfor maskinlæring feltet. Det finnes eksempler på kode på nett som bruker maskinlæring til nettopp dette, både som enkeltstående maskinlæring kode, og applikasjoner som implementerer det med en android applikasjon.

Alternative GPS-løsninger som henter inn veginformasjon, og deretter tilbyr dette i brukergrensesnitt, er ofte knyttet opp mot bilprodusenter sine egne systemer. Vi finner derimot ingen tilsvarende alternativer som lar uavhengige brukere frivillig innhente data til bruk av fellesskapet i open source prosjekt.

## 4. Produktoversikt

### 4.1 Forutsetninger og avhengigheter

- Tilstrekkelig med treningsdata for trafikkskilt.
- Tilgang til bil for å bruke applikasjonen, evt elektrisk sparkesykkel / buss
- Tilgang til android mobil med brukbart kamera

### 5. Produktets funksjonelle egenskaper

Beskrivelse
Bruke kamera for å filme trafikkskilt mens brukeren kjører
Lagring av innhentet data til ekstern server
Modell som bruker innhentet data for trenings
Trenings av innhentet data for klassifisering av trafikkskilt
Brukeren skal kunne få tilbakemelding av appen i sanntid, hva fartsgrensen er
Appen skal lagre trafikkskilt med geolokasjon, for bruk av OpenStreetMap
Grafisk brukergrensesnitt som viser trafikkskilt i OpenStreetMap
Kartvisning med posisjon i sanntid
Innsending av data til OpenStreetMap

---

## Vedlegg

### Vision Document

#### **6. Ikke-funksjonelle egenskaper og andre krav.**

- Den mobile applikasjonen skal være tilgjengelig på Android 5.0 eller nyere
- Løsningen skal være i samsvar med WCAG 2.1 prinsipp 1 (Begripelig), og ha god brukskvalitet.
- API-et skal være dokumentert i kode, og på prosjektet sin WIKI
- Programmet skal bruke OpenCV for tolkning av trafikkskilt
- Android-applikasjonen skal utvikles i Android Studios, i Kotlin eller Java.
- Programmet skal kunne brukes med OpenStreetMap som kart api.

---

## Appendix

### Tidsforbruk perioder på halve måneder



Figur 18: Timefordeling per halve måned