



Sjøkrigsskolen

Bacheloroppgave

Maskinlæring som klassifiseringsverktøy – den nye Marinen?

av

Marthe Engvik og Erlend Omland

Levert som en del av kravet til graden:

BACHELOR I MILITÆRE STUDIER MED FORDYPNING I ELEKTRONIKK OG DATA

Innlevert: desember 2018

Godkjent for offentlig publisering

Publiseringsavtale

En avtale om elektronisk publisering av bachelor/prosjektoppgave

Kadettene har opphavsrett til oppgaven, inkludert rettighetene til å publisere den.

Alle oppgaver som oppfyller kravene til publisering vil bli registrert og publisert i Bibsys Brage når kadettene har godkjent publisering.

Oppgaver som er graderte eller begrenset av en inngått avtale vil ikke bli publisert.

Vi gir herved Sjøkrigsskolen rett til å gjøre denne oppgaven tilgjengelig elektronisk, gratis og uten kostnader	<input checked="" type="checkbox"/> Ja	<input type="checkbox"/> Nei
Finnes det en avtale om forsinket eller kun intern publisering? (Utfyllende opplysninger må fylles ut)	<input type="checkbox"/> Ja	<input checked="" type="checkbox"/> Nei
Hvis ja: kan oppgaven publiseres elektronisk når embargoperioden utløper?	<input type="checkbox"/> Ja	<input type="checkbox"/> Nei

Plagiaterklæring

Vi erklærer herved at oppgaven er vårt eget arbeid og med bruk av riktig kildehenvisning. Vi har ikke nyttet annen hjelp enn det som er beskrevet i oppgaven.

Vi er klar over at brudd på dette vil føre til avvisning av oppgaven.

Dato: 04.12.2018

Erlend Austad Omland

Kadett navn

Kadett, signatur

Marthe Lodding Engvik

Kadett navn

Kadett, signatur

Forord

Denne bacheloroppgaven er skrevet som et krav for bachelor i militære studier med fordypning i data og elektronikk, og er skrevet i perioden mai 2018 til desember 2018.

Oppgaven har gitt oss muligheten til å lære mer om maskinlæring, samtidig som den har gitt oss et innblikk i hva autonome systemer kan brukes til i Marinens. Det er i denne oppgaven lagt vekt på å diskutere behovet for et autonomt deteksjon- og klassifiseringssystem. I tillegg til dette fremstilles også vurderinger av det tekniske arbeidet som er lagt til grunn for å skape et faktisk system som nettopp kan gjenkjenne objekter på havoverflaten.

Vi ønsker å takke våre veiledere, forsker Martin Vonheim Larsen og førsteamanuensis Christophe Massacand, for hjelp og veiledning gjennom perioden. Videre vil vi takke orlogskaptein Frode Mjelde og førsteamanuensis Alexander Sauter for ideer og innspill. Også løytnant Bjarne Vågenes og løytnant Øyvind Eide fortjener en takk for bidrag og tilrettelegging for demonstrasjon og distribusjon av intervuspørsmål.

Bergen, Sjøkrigsskolen, 04.12.2018

Marthe Lodding Engvik

Erlend Austad Omland

Oppgaveformulering

På bakgrunn av en stor utvikling innenfor autonomi har Forsvaret og forsvarsindustrien fått øynene opp for de muligheter som autonome systemer har å by på. I vår egen langtidsplan står det følgende: "Forsvaret skal utvikle seg videre mot et høyteknologisk forsvar, og sentrale forsknings- og utviklingsområder (FoU-områder) vil være informasjons- og nettverksteknologi, missilteknologi, autonome og ubemannede systemer og forsvar mot den samme nye teknologien" (Forsvarsdepartementet, 2015, 101).

Maskinlæring spiller en stor rolle i utviklingen av autonome systemer. En viktig del av dette er utviklingen av gjenkjennings- og klassifiseringssystemer for bruk til analyse av visuell data. Dette brukes til å bygge automatiserte systemer som kan gjennomføre oppgaver på en like god, og gjerne bedre, måte enn menneskets visuelle system (He, Zhang, Ren & Sun, 2015). Mange maskinlæringsystemer er relatert til utvinning av informasjon fra 3D- og 2D-data som innhentes av kameraer. Et menneske er laget for å innhente informasjon, prosessere den, prøve å forstå dynamikken og videre ta beslutninger basert på visuell persepsjon (Huang, 1996). Dette er også det en med maskinlæring prøver å gjenskape med en datamaskin ved bruk av algoritmer. Hvorvidt slike systemer kan bidra til å øke Sjøforsvarets operative evne er fortsatt uvisst.

Med bakgrunn i dette har vi i denne oppgaven sett nærmere på temaet;

Maskinlæring som klassifiseringsverktøy – den nye Marinen?

Vi har valgt i denne oppgaven å se på temaet ved å utvikle et eget klassifiseringssystem basert på maskinlæring, samt introdusere dette produktet for ansatte i Marinen. Produktet har først og fremst blitt brukt til å skape forståelse for og innsikt i maskinlæring og autonomi. Videre har vi brukt vårt produkt til å skape et bilde av hva et slikt system, og liknende systemer, er kapable til. Med dette som grunnlag drøfter vi hvorvidt et autonomt gjenkjenningsssystem kan nytties i dagens Marine og i hvilke kapabiliteter som må til for at dette skal være en ressurs i fremtidig oppdragsløsning og beslutningstaking.

Sammendrag

I Forsvaret er det et økt fokus på innfasing av autonome og ubemannede systemer (Forsvarsdepartementet, 2015, 101). Et fagfelt som bidrar til utvikling av autonome systemer er maskinlæring som er et underfelt av kunstig intelligens. Fagfeltet søker å gi maskiner evnen til å lære, samt å detektere og klassifisere objekter i omgivelsene rundt seg, noe teknologien i flere sammenhenger har vist seg å gjøre bedre enn mennesker (He et al., 2015).

I denne oppgaven har vi utviklet et klassifiseringssystem ved å bruke maskinlæring med et mål om å undersøke om teknologien kan anvendes til å automatisere, overta eller avlaste oppgaver som i dag gjøres av operatører i Marinen. Videre har vi gjennomført en demonstrasjon av systemet for erfarte ansatte i Sjøforsvaret. I forbindelse med demonstrasjonen ble det gjennomført flere intervjuer for å avdekke hvilke områder det kan tenkes at et klassifiserings-system kan gi merverdi i operativ sammenheng.

Intervjuene avdekket at det vil være behjelplig med et system som kan bidra til hurtigere og mer sikker visuell deteksjon og klassifisering. Dette for å øke beslutningsgrunnlaget, samt effektivisere beslutningsprosessen. Et maskinlæringssystem kan også bidra til å avlaste personell og i tillegg ivareta kompetanse om bord.

De tekniske resultatene viser at systemet vi har laget produserer gode resultater innenfor oppgavebegrensningene med relativt høy konfidens. Det skal dog nevnes at systemet presterer dårlig på de bildene som er utelatt fra treningsgrunnlaget på bakgrunn av oppgavebegrensningen. Dette eksemplifiserer det faktum at store og varierte datamengder er essensielt innen maskinlæring, og spesielt innenfor klassifisering, for å oppnå høy pålitelighet.

På bakgrunn av oppgavens tema og de resultatene vi har oppnådd, konkluderer vi med at maskinlæring som teknologi er moden for operativ bruk og er et fagområde som Forsvaret bør satse på. Videre har våre tekniske tester bekreftet viktigheten av datainnsamling, og vi anbefaler derfor at Forsvaret i første omgang implementerer prosedyrer for storskala datainnsamling og annotering til maskinlæringsformål.

Innholdsfortegnelse

Figurer	3
Tabeller	4
Nomenklatur	5
1 Innledning	6
1.1 Bakgrunn	6
1.2 Mål	8
1.3 Begrensninger	8
1.4 Metode	9
1.4.1 Metode for systemdesign	9
1.4.2 Demonstrasjon og intervju	11
2 Teori	12
2.1 Autonomi og automatisering	12
2.2 Maskinsyn	13
2.3 Nevrale konvolusjonsnettverk	14
2.4 Nettverkssarkitektur	17
2.5 Trening	19
3 Systemdesign	23
3.1 Maskinvare	24
3.2 Programvare	25
3.3 Konfigurasjon av YOLOv3	26
3.4 Treningsgrunnlag	26
3.5 Validering og bruk	29
4 Resultater	30
4.1 Testing av klassifiseringssystemet	30
4.2 Demonstrasjon og intervju	34
5 Drøfting	36
5.1 Klassifiseringssystemet	36
5.2 Maskinlæring som klassifiseringssystem	40
6 Konklusjon med anbefaling	44
Bibliografi	46
Vedlegg	49

Vedlegg A – Trening	49
A.1 - Trening 1	49
A.2 - Trening 2	52
A.3 - Trening 3	54
A.4 - Trening 4	57
Vedlegg B – Python Kildekode.....	60
B.1 – CLR.py.....	60
B.2 - create_train-test.py	61
B.3 - plot_mApy	62
B.4 - flipImages.py	63
B.5 - rename.py.....	64
B.6 - imgaug.py	65
Vedlegg C – CMake og C++ Kildekode	66
C.1 – CMakeLists.txt	66
C.2 – main.cpp.....	67
C.3 – funksjoner.h.....	68
Vedlegg D – Konfigurasjonsfiler	70
D.1 - obj.data	70
D.2 - obj.names.....	70
D.3 - test.txt	70
D.4 - train.txt	70
D.5 - yolov3-Utkikk9000.cfg	70
Vedlegg E – Datasett.....	70
Vedlegg F – Vektfiler	70
F.1 - v1.0	70
F.2 - v2.0	70
F.3 - v3.0	70
F.4 - v4.0	70
Vedlegg G – Videofiler fra testing.....	71
G.1 - Forhold_dårlige.avi.....	71
G.2 - Forhold_optimale.avi	71
Vedlegg H – Åpen-kildekode	71

Figurer

Figur 1 - 18x18 pikslers bilde sett av menneske og datamaskin (Geigetry, 2014)	13
Figur 2 - Kunstig nevralt nettverk (Nielsen, 2018, «Using neural nets..»).....	14
Figur 3 - Konvolusjonsprosessen	15
Figur 4 - Illustrasjon vekter.....	16
Figur 5 - Maxpooling.....	16
Figur 6 - Darknet 53 (Redmond & Farhadi, 2018, 2).....	17
Figur 7 - Residuale koblinger i YOLOv3 (Kathuria, 2018).....	17
Figur 8 – Eksempel på Darknet annoteringsfil.....	19
Figur 9 - Forskjellig tilpassede modeller (Goodfellow et al., 2016, 112).....	21
Figur 10 – Generaliseringsgap (Goodfellow et al., 2016, 113).....	22
Figur 11 - Systemtegning.....	23
Figur 12 - Prosessen ved produsering av datasett	27
Figur 13 - Skjermbilde fra annoteringsprosessen.....	27
Figur 14 - mAP valideringsdata (fase 3) og validerings- og treningsdata (fase 4)	30
Figur 15 - Test gjennom periskop. Her med konfidens 100%, 64% og 96%	32
Figur 16 - Dårlige lysforhold (oppe venstre), optimale forhold (oppe høyre) og høy sjø (to nederste bilder).....	33
Figur 17 - Loss-chart til venstre og mAP til høyre.....	50
Figur 18 - Loss-chart, random = 0.....	53
Figur 19 - mAP med random = 0 til venstre og mAP med random = 1 til høyre	53
Figur 20 - Loss-chart	55
Figur 21 - mAP med initialiserte vekter til venstre og mAP med egne vekter til høyre	55
Figur 22 - Detaljerte mAP-data fra treningssett	56
Figur 23 - Loss-chart	58
Figur 24 - mAP og generaliseringsgap	58
Figur 25 - Detaljert mAP for treningssett til venstre og for valideringssett til høyre	59

Tabeller

Tabell 1- Kravsanalyse	9
Tabell 2 - Definerte grader av autonomi, oversatt (Sheridan & Verplank, 1978, 8-17). .	12
Tabell 3 - Maskinvare.....	24
Tabell 4 - Ulike modellers prestasjon etter testing.....	25
Tabell 5 - Utvikling av presisjon.....	31
Tabell 6 - Datasett trening.....	49
Tabell 7 - Datasett trening 2.....	52
Tabell 8 - Datasett trening 3.....	54
Tabell 9 - Datasett trening 4.....	57

Nomenklatur

API	Application Programming Interface
Batch	Antall datasampler som lastes inn i VRAM hver treningsiterasjon
C	Lavnivå programmeringsspråk
CUDA	Compute Unified Device Architecture, API for å bruke GPU til generell dataprosessering
cuDNN	CUDA Deep Neural Network, programmeringsbibliotek for dyp læring som bruker CUDA
FoU	Forskning- og utviklingsarbeid
FPS	Frames per second
GPU	Graphics Processing Unit
IKT	Informasjons- og kommunikasjonsteknologi
IR	Infrarød
mAP	Mean Average Precision
OpenCV	Programmeringsbibliotek for maskinsyn
OS	Operativsystem
RAM	Random Access Memory
RGB	Red-Green-Blue
VRAM	Video Random Access Memory, GPUens minne

1 Innledning

Denne oppgaven tar for seg hvordan dagens moderne maskinlæringsteknologi kan anvendes i militær sammenheng. Vi ser konkret på hvordan et system for objektgjenkjenning kan produseres for å tjene en operasjonell hensikt, samt hvilke implikasjoner vårt system, og tenkte liknende systemer kan ha for fremtidens Marine.

Oppgaven er delt inn i seks hovedkapitler. I innledningen presenteres bakgrunn, mål og metode for oppgaven. Det neste kapittelet består av nødvendig teori om maskinlæring og nevrale nettverk. Videre presenteres designet og utarbeidelsen av systemet i kapittel tre. I kapittel fire fremlegges det resultater fra tester gjort under utarbeidelsen av klassifiseringssystemet, samt resultater fra praktisk demonstrasjon og påfølgende intervjuer av ansatte i Marinen. Kapittel fem tar for seg drøfting rundt de tekniske valgene gjort under den praktiske utarbeidelsen av systemet, samt bruken av et automatisk gjenkjennelsessystem i et operativt miljø. Til slutt fremkommer konklusjon med anbefaling i kapittel seks.

1.1 Bakgrunn

De siste år har verden sett store fremskritt innen automatisering av forskjellige funksjoner gjennom datamaskiner. Språkforståelse, objektklassifisering, storskala dataanalyse og flere andre bruksområder har blitt implementert i datamaskiner, og forskjellen mellom hva et menneske og en maskin kan gjøre begynner å minke. Mange av oppgavene Forsvaret løser innehar et element av analyse og overvåkning. I Marinen er en av oppgavene å drive nettopp overvåking av norsk territorialfarvann. Herunder er klassifisering av overflatefartøy en essensiell del av overvåkingen, noe som maskiner potensielt er bedre rustet til å gjennomføre enn mennesker. Til tross for dette er foreløpig denne typen teknologi ikke i bruk i stor skala, men Forsvarets fremtidsplaner virker å sikte seg inn mot mer satsning på teknologien. I Forsvaret er det allerede besluttet at autonome og ubemannede systemer skal innfases (Forsvarsdepartementet, 2015, 101). I den siste utgaven av NECESSSE fra høsten 2018 skrives det at ”operativ anvendelse av ubemannede systemer kan forbedre situasjonsforståelsen, redusere menneskelig arbeidsbelastning, og minimere faren for tap og skade på sivilt og militært personell” (Hareide et al., 2018, 137). Autonome systemer vil altså kunne være et bidrag innenfor flere områder.

Eksempler på dette vil være bruk av autonome enheter under risikofylte oppdrag for å minnere sjansen for tap av menneskeliv, slik som Odin.¹ Videre er det mulighet for at slike systemer også vil kunne bidra til effektivisering av beslutningstaking, samt gi avlastning i en allerede ”lean manned” marine.

Maskinlæring, et underfelt av kunstig intelligens, har i nyere tid vært under stor utvikling. Fagfeltet søker å gi maskiner evnen til å lære, samt å detektere og klassifisere objekter i omgivelsene rundt seg, noe teknologien i flere sammenhenger har vist seg å gjøre bedre enn mennesker (He et al., 2015). Med hastigheten som objektklassifisering forbedres i, er det mulig at teknologi som maskinlæring kan levere klassifikasjon med høy nok konfidens til at automatisk målangivelse og navigasjon snart kan være en realitet. De siste års utvikling innenfor objektklassifisering har vekket interessen for å undersøke om slik teknologi kan gi merverdi i Marinens. Hvorvidt dette er teknologi for fremtidens norske Marine er bakgrunnen for det som har utviklet seg til oppgavens tema.

¹ Odin – ubemannet minerydderfarkost

1.2 Mål

I denne oppgaven har vi utviklet et klassifiseringssystem ved å bruke maskinlæring med et overordnet mål om å undersøke om teknologien kan anvendes til å automatisere, overta eller avlaste oppgaver som i dag gjøres av operatører i Marinen.

1.3 Begrensninger

Maskinlæring har utallige bruksområder, men vi har i denne oppgaven valgt å begrense oss til objektklassifisering med fartøy som plattform for et slikt klassifiseringssystem. Som en følge av dette har vi begrenset oss til at systemet skal levere en enkel gjenkjenning av overflatefartøyer med relativ kurs. Mange av begrensningene tilknyttet produktet kommer som en følge av tid, da utvikling av et maskinlæringssystem er tidkrevende. Vi har valgt å begrense系统的 treningsgrunnlag til åpne kilder, dermed vil gjenkenningskapabiliteten være av mer konseptuell nyttighet. Dette har likevel bidratt til å gi et inntrykk av hva liknende systemer faktisk kan utrette i fremtidens norske Marine. En annen begrensning i vår oppgave omfatter drøfting rundt de etiske spørsmålene som reises ved bruk av autonome systemer i en operativ setting. Dette er et aspekt vi er bevisste over, men ikke har gjort rede for i oppgaven. Drøftingen er begrenset til diskusjon rundt de tekniske mulighetene for et klassifiseringssystem i Marinen, og utelater med dette undersøkelser rundt en faktisk implementasjon av et slikt system.

1.4 Metode

Dette delkapittelet har til hensikt å beskrive metode for fremgangsmåten i utviklingen av klassifiseringssystemet som er laget i forbindelse med denne oppgaven. Videre presenteres metoden for demonstrasjonen av klassifiseringssystemet og påfølgende intervju.

1.4.1 Metode for systemdesign

I militær, operativ sammenheng, er tid en begrensende faktor. Både på bro og i operasjonsrom må avgjørelser tas hurtig og informasjon blir etter kort tid foreldet. Videre er det viktig at informasjonen som presenteres er representativ for virkeligheten, dette for å sikre gode operasjonelle avgjørelser. På bakgrunn av dette bestemte vi oss tidlig for noen krav, vist i tabell 1, vi ønsket at valgt maskinvare og programvare skulle møte for at produktet skulle kunne anvendes som et verktøy i Marinens.

Tid har også vært en begrensende faktor i utarbeidelsen av oppgaven. For å kunne utarbeide et produkt innenfor gitt tidsramme til arbeidet med oppgaven, har vi i stor grad benyttet oss av programvare som er åpen-kildekode. Videre har det også vært nødvendig å bruke denne type programvare som en følge av at maskinlæring ikke er en del av pensum på Sjøkrigsskolen.

FAKTOR	BEGRENSENDE EFFEKT	ØNSKET SLUTTIL-STAND	KRAV
TID	Foreldet informasjon kan føre til et virkelighetsbilde som ikke representerer virkeligheten.	Prosessere bilder i sanntid for å kunne nytte informasjonen til operasjonell handling.	Prosesseringshastighet > 20 Frames Per Second (FPS)
	Oppgavens rammer med hensyn på tid, begrenser hva vi kan utvikle.	Spare tid på programvareutvikling.	Programvare må være åpen-kildekode
PRESISJON	Upresise deteksjoner kan føre til feilaktig informasjon	Legge til rette for nøyaktig beslutningsgrunnlag.	Mean Average Precision (mAP) > 80%

Tabell 1- Kravsanalyse

En stor del av fremgangsmetoden i systemutarbeidelsen har vært nært knyttet til produksjonen av vårt eget datasett. Dette arbeidet har vært styrt av kravsanalysen, med tanke på kravene vi har satt til presisjon. Resultatet har blitt en fremgangsmetode som har delt inn data-innsamling og produksjon i fire faser, som blir nærmere forklart i avsnittet om systemdesign.

Fase 1: Web-scraping

Fase 2: Data-augmentering ved speiling

Fase 3: Negative dataeksempler

Fase 4: Data-augmentering ved utklipp

Produksjonen av dette datasettet er potensielt et meget tidkrevende prosjekt, og vi har av denne årsak forsøkt å automatisere mest mulig av arbeidet.

1.4.2 Demonstrasjon og intervju

For å kunne knytte vårt selvutviklede system til operasjonelt bruk av autonomi i Marinen valgte vi å gjennomføre en demonstrasjon for erfarne representanter fra ulike avdelinger. Vi hadde et ønske om å komme i kontakt med erfarne ansatte fra ulike avdelinger i Marinen, og fikk etter forespørsel fra skolen en liste over potensielle representanter. Utvalget var tilfeldig i den grad av at alle vi henvendte oss til, ikke nødvendigvis kunne møte for demonstrasjon og intervju.

Demonstrasjonen gikk ut på å vise vårt system og dets prestasjon på realistiske omgivelser, før vi deretter snakket om hvor et slikt system kan tenkes å benyttes ute på fartøy. For å videre undersøke tankesett og holdninger til å implementere et autonomt system om bord gjennomførte vi også et e-postintervju med representantene som var til stede under demonstrasjonen.

Intervjuet skulle sørge for at vi fikk mer utfyllende svar fra erfarne ansatte rundt temaet autonomi i Marinen. Vi valgte å gjennomføre intervjuet per e-post fordi vi ønsket svar like etter demonstrasjonen. En slik intervjuype ble også valgt fordi vi ønsket å sikre svar fra alle representert på demonstrasjonene i den hensikt å se om det var varierende behov og holdninger i de ulike avdelingene. Videre tenkte vi at et e-postintervju ville gjøre det enklere for oss å kunne benytte svarene i senere drøfting. Intervjuet ble lagt opp som følger;

Intervjuspørsmål:

Etter å ha deltatt på demonstrasjon av vårt produkt til vår bacheloroppgave har vi noen spørsmål vi håper du kan ta deg tid til å besvare. Vi ønsker så utfyllende svar som mulig slik at dette er synspunkter og betrakninger vi kan bruke til en videre analyse i vår oppgave.

1. Hvilke fordeler/ulemper kunne vårt produkt, eller lignende systemer, gitt din avdeling?
2. Hva er holdningene og tankene rundt bruk av slike systemer i din avdeling?
3. Hvilke kvaliteter/egenskaper ville du at systemet skulle hatt dersom det skulle erstattet en bemannet stilling?
4. Hvilke oppgaver har du tillit til at slike systemer skal kunne løse?

2 Teori

Dette kapittelet tar for seg nødvendig teori for oppgaven. Hoveddelen av teorien består av begrepsavklaringer og informasjon som er nødvendig for å kunne sette seg inn i hva som er blitt gjort i forbindelse med utvikling av klassifiseringssystemet, som er laget for denne bacheloroppgaven.

2.1 Autonomi og automatisering

Autonomi betyr delvis eller fullstendig selvstendighet, selvstyre eller selvbestemmelse (Sagdahl, 2017). Altså snakker vi her om systemer som kan ta beslutninger og utføre handlinger uten menneskelig inngripen. Til tross for at en ofte snakker om autonomi som selvstyrte maskiner er det likevel ingen felles definisjon for autonomi, og begrepet brukes om systemer som er alt fra delvis fjernstyrte til fullstendig autonome. For å kaste lys på hva autonomi kan bety kan en benytte seg av Sheridan og Verplank sin tabell for ti grader av autonomi. Tabellen går fra et nivå hvor mennesket gjør alt til et nivå hvor datamaskinen tar alle avgjørelser (Sheridan & Verplank, 1978, 8-17).

NIVÅ	BESKRIVELSE
1	Ingen assistanse, mennesket tar alle beslutninger og systemet bare utfører
2	Datamaskinen tilbyr en komplett oversikt over beslutningsalternativene
3	Datamaskinen foreslår beslutningsalternativer
4	Datamaskinen foreslår ett beslutningsalternativ, mennesket bestemmer om det skal utføres
5	Datamaskinen utfører foreslått alternativ hvis godkjent av menneske
6	Datamaskinen tillater mennesket veto i en begrenset tid før utførelse
7	Datamaskinen utfører automatisk og informerer mennesket
8	Datamaskinen utfører automatisk, informerer mennesket bare ved forespørsel
9	Datamaskinen utfører automatisk, informerer mennesket bare hvis forhåndsprogrammert
10	Datamaskinen bestemmer alt, uten menneskelig innblanding

Tabell 2 - Definerte grader av autonomi, oversatt (Sheridan & Verplank, 1978, 8-17)

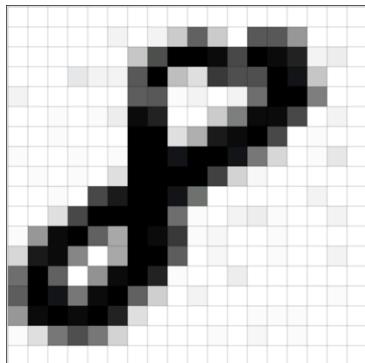
I denne oppgaven har vi ikke utviklet et system som utfører en handling, som tabellen gir utsyn for, men en analyse. På bakgrunn av dette vil vi ikke kunne klassifisere vårt produkt ved bruk av Sheridan og Verplank sin tabell. Likevel ser vi for oss at ved en implementering av vårt system vil dette kunne bidra til et automatisert system som er av grad tre eller fire i tabell 2. En slik implementering baserer seg på et system som foreslår alternativ(er), med en viss konfidens, for operatøren og lar operatøren selv utføre handlingen basert på informasjon fra systemet. Vårt system vil i denne sammenhengen være et verktøy som bidrar med informasjon til beslutningstaking.

2.2 Maskinsyn

Computer vision is the transformation of data from a still or video camera into either a decision or a new representation. All such transformations are done for achieving some particular goal (Bradski & Kaehler, 2008, 2).

Datamaskiner kan av natur ikke forstå bildeinformasjon på et dypere nivå enn som en ansamling av data. Maskinsyn er et fagfelt som sikter mot å gi datamaskiner muligheten til å se og oppfatte sine omgivelser. Dagens mest effektive metode for å oppnå dette er å bruke nevrale nettverk, og mer spesifikt nevrale konvolusjonsnettverk som anvender den matematiske operasjonen konvolusjon til å behandle data.

Datamaskiner oppfatter bilder annerledes enn mennesker, som illustrert i figur 1. Det som mennesker ser som et bilde av et åttetall, ser en datamaskin som en matrise av intensitetsverdier, som vist til høyre i figur 1. Målet med maskinsyn er å oversette slike intensitetskort til klassifisering av objekter.

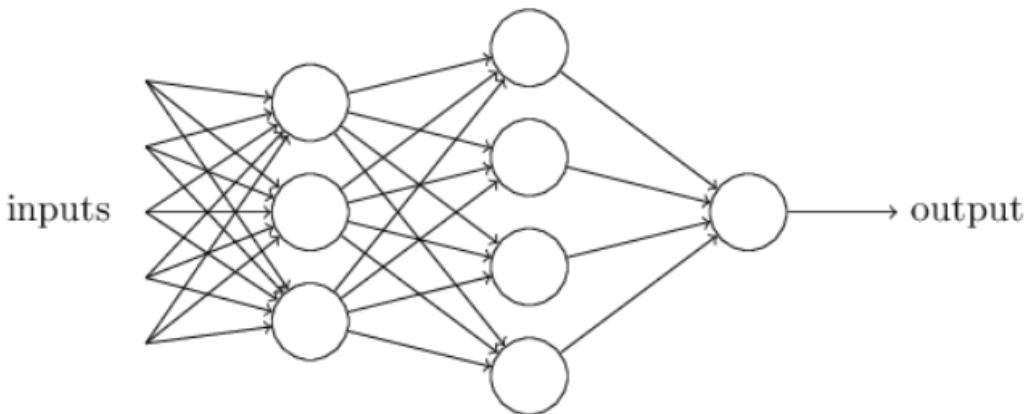


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	253	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	253	255	254	254	216	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figur 1 - 18x18 pikslers bilde sett av menneske og datamaskin (Geigetry, 2014)

2.3 Nevrale konvolusjonsnettverk

Konseptene for nevrale nettverk har eksistert siden 1944, men de siste års fremgang med regnekraften til moderne Graphics Processing Units (GPUer) har medført at konseptet nå er moden for kommersielt og forskningsbasert bruk (Hardesty, 2017). Et nevralt nettverk er i sin enkleste form en ansamling av nevroner, implementert som matematiske funksjoner organisert i flere lag. Nevronene gir en output avhengig av sine inputs og en matematisk funksjon som knytter disse sammen. Outputen fra det siste laget brukes til å gjøre en prediksjon på hva dataene en sender inn i nettverket representerer.



Figur 2 - Kunstig nevralt nettverk (Nielsen, 2018, «Using neural nets...»)

En spesiell type nevralt nettverk som brukes i maskinsyn kalles konvolusjonsnettverk. Dette er et nevralt nettverk som er spesielt godt egnet til å behandle data som er organisert i en todimensjonal gitterstruktur (Goodfellow, Bengio & Courville, 2016, 326). Et konvolusjonsnettverk har et eller flere lag hvor en konvolusjonsoperasjon brukes for å behandle data. I motsetning til mer tradisjonelle kunstige nevrale nettverk, hvor alle inputnevronene er koblet til alle outputnevronene, har konvolusjonsnettverk betydelig færre koblinger. En stor fordel med disse nettverkene, som er av stor betydning for denne oppgaven, er antallet parametere som skal trenes. Konvolusjonsnettverket har relativt få parametere og som en følge av dette er det derfor færre operasjoner som må gjennomføres når data skal behandles av nettverket (Goodfellow et al., 2016, 331). Dette medfører at vi, ved å bruke en konvolusjonsarkitektur, kan behandle data med høyere hastighet enn om vi hadde brukt et vanlig nevralt nettverk. For å produsere informasjon som skal kunne anvendes operasjonelt er hurtighet viktig, og derfor er en slik arkitektur fordelaktig å bruke i denne oppgaven. En ytterligere fordel med det relativt lave antallet parametre som skal trenes er lavere risiko for overtrenings. Dette kan knyttes opp mot det faktum at færre trenbare parametere resulterer i at modellen blir tvunget til å generalisere i større grad, enn et klassisk nevralt nettverk, med «fully connected» lag.

Konvolusjon er en matematisk operasjon over flere funksjoner der outputfunksjonen er sammen av konvolusjonen mellom en input og et filter. Siden konvolusjon innen maskinlæring ofte er en konvolusjon mellom flerdimensjonale matriser, brukes som oftest kryss-korrelasjonsformelen vist i formel 1. I denne formelen er $S(i,j)$ resultantmatrisen, $I(m,n)$ er inputmatrisen, og $K(m,n)$ er filteret. Bokstavene i og j er indekser i resultantmatrisen, og m og n er indekser i inputmatrisen og filteret.

$$S(i,j) = (K*I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m, n) \quad (1)$$

Innenfor maskinlæring kalles denne operasjonen ofte for konvolusjon, og når vi bruker begrepet konvolusjon i denne oppgaven refererer vi også til formel 1. (Goodfellow et al., 2016, 329). Rent matematisk er operasjonene veldig like. Figur tre viser hvordan konvolusjonsoperasjonen fungerer i maskinlæring, og det er denne metoden vi kommer til å referere til som konvolusjon i denne oppgaven.

Konvolusjonen fungerer ved at en legger et filter, som vist øverst i høyre hjørne av figur 3, over bilder som skal konvoleres. Verdiene i filteret multiplisieres med verdiene i bildet, produktene summeres, og summen er en ny pikselverdi i resultantmatrisen som konvolusjonsoperasjonen produserer. For figur 3, vil den første pikselverdien bli:

$$3 * (1 * 40) + 3 * (0 * 40) + 3 * (-1 * 40) = 0$$

Filteret vil deretter flyttes en kolonne mot høyre, og neste nye pikselverdi blir:

$$3 * (1 * 40) + 3 * (0 * 40) + 3 * (-1 * 0) = 120$$

Filteret i figur 3 er et enkelt filter for deteksjon av vertikale skiller i bilder. Nevlale konvolusjonsnettverk har tussenvis av forskjellige filtre, som skal detektere forskjellige karakteristikker i et bilde. Om de riktige trekkene detekteres i bildet, vil nettverkets deteksjonsslag gi en output om at et objekt er tilstede i bildet (Ng, 2017a). Verdiene i filtermatrisene er det vi kaller vekter, og det er disse vektene vi tilpasser når vi trener et nettverk.

1	0	-1
1	0	-1
1	0	-1

40	40	40	0	0	0
40	40	40	0	0	0
40	40	40	0	0	0
40	40	40	0	0	0
40	40	40	0	0	0
40	40	40	0	0	0

Figur 3 - Konvolusjonsprosessen

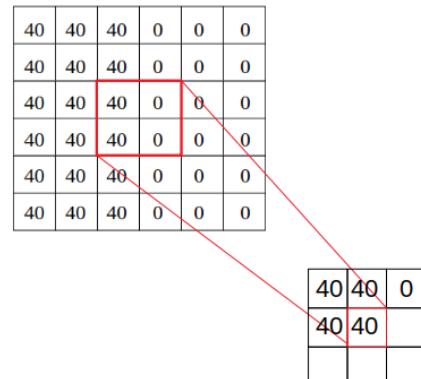
W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

**Figur 4 - Illustrasjon
vekter**

Forskning gjort av Matthew Zeiler og Rob Fergus ved New York University viser at de første lagene i et konvolusjonsnett lærer seg å gjenkjenne enkle former, og at kompleksiteten i strukturene som gjenkjennes, øker i takt med dybden på nettverket (Zeiler & Fergus, 2013, 5). Denne artikkelen viser tydelig kompleksiteten i de strukturer som dypere lag i konvolusjonsnett oppnår høy aktivering for, samt hvordan denne kompleksiteten øker med nettverksdybde.

For å minimere prosesseringstiden i databehandlingen bruker konvolusjonsnettverk forskjellige funksjoner for pooling. Disse funksjonene bidrar til å konsentrere og minske mengden data som forflyttes i nettverket, samt data som skal lastes inn i GPUens Video Random Access Memory (VRAM). Videre fører pooling til at nettverket oppnår en viss grad av invarians til forskjellige posisjoner og orienteringer på objekter. Dette betyr at nettverket kan gjenkjenne objekter av samme type, uavhengig av om objektet er annerledes orientert eller posisjonert i et bilde i forhold til treningsgrunnlaget (Dettmers, 2018).

Maxpooling er ofte brukt, og er illustrert i figur 5. Funksjonen tar vare på den høyeste pikselverdien som et filter ser, og prosesserer et helt bilde på samme måte som et filter i konvolusjon. Averagepooling, som brukes i YOLOv3, tar gjennomsnittet av pikslene i stedet for høyeste verdi.



Figur 5 - Maxpooling

2.4 Nettverkssarkitektur

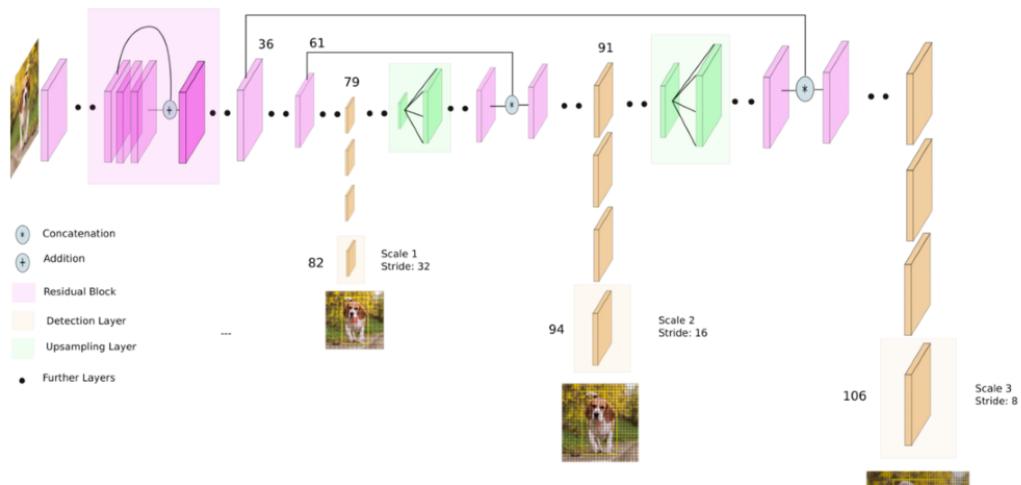
Nettverket vi har valgt å benytte for å klassifisere objekter i oppgaven, YOLOv3, bruker en underliggende arkitektur, Darknet-53. Dette er en “feature extractor” som henter ut informasjon om karakteristikker ved et bilde. Videre har arkitekturen 53 ytterligere lag som gjør selve deteksjonsjobben (Redmond & Farhadi, 2018).

Totalt benytter denne nettverksarkitekturen 106 lag, som er relativt dypt. Styrken til YOLOv3 er at presisjonen nettverket leverer er sammenlignbart med dypere nettverk som *ResNet-152*, men i dobbelt så høy hastighet (Hui, 2018).

Type	Filters	Size	Output
Convolutional	32	3×3	256 x 256
Convolutional	64	$3 \times 3 / 2$	128 x 128
1x	32	1×1	
Convolutional	64	3×3	
Residual			128 x 128
Convolutional	128	$3 \times 3 / 2$	64 x 64
Convolutional	64	1×1	
2x	128	3×3	
Residual			64 x 64
Convolutional	256	$3 \times 3 / 2$	32 x 32
Convolutional	128	1×1	
8x	256	3×3	
Residual			32 x 32
Convolutional	512	$3 \times 3 / 2$	16 x 16
Convolutional	256	1×1	
8x	512	3×3	
Residual			16 x 16
Convolutional	1024	$3 \times 3 / 2$	8 x 8
Convolutional	512	1×1	
4x	1024	3×3	
Residual			8 x 8
Avgpool		Global	
Connected		1000	
Softmax			

Figur 6 - Darknet 53 (Redmond & Farhadi, 2018, 2)

Et problem med dype konvolusjonsnettverk er at jo dypere i nettverket en kommer, jo mer utvannet blir informasjonen om karakteristikkene i bildet, som en følge av mange konvolusjoner. Empirisk forskning på dype nevrale konvolusjonsnettverk bekrefter dette (Ng, 2017b). Løsningen som YOLOv3, og andre dype konvolusjonsnettverk anvender, er residuale koblinger. Denne metoden kobler sammen informasjon om karakteristikker ved objekter fra tidlige stadier i nettverket, og konrollerer det sammen med informasjonen som er dypere i nettverksarkitekturen i det som kalles en residual blokk. Nettverket bruker så denne informasjonen til å gjøre deteksjoner i forskjellige skalaer, dypere i nettverket.



YOLO v3 network Architecture

Figur 7 - Residuale koblinger i YOLOv3 (Kathuria, 2018)

YOLOv3 detekterer objekter i bilder i tre forskjellige deteksjonslag, vist som de gule lagene i figur 7. Hvert lag foretar seg deteksjoner av objekter i økende skala, fra små til store objekter. De residuale koblingene er illustrert i form av linjer mellom tidlige og senere lag, eksempelvis som mellom lag 36 og etter oppskaleringen bak lag 91, vist i figur 7.

Konfidens

YOLOv3, og andre objektklassifiseringssystemer, produserer output i form av en boks rundt objekter sammen med en konfidens på hva objektet er. Hver gang YOLOv3 gjør en prediksjon på hvilket objekt som er i bildet, gjør nettverket både objektdeteksjon og objektklassifisering samtidig. Deteksjonen i prediksjonen forteller oss hvor sikkert nettverket er på at et objekt er tilstede i den predikerte boksen, gitt ved midtpunkt, bredde og høyde (Redmond, Divvala, Girshik & Farhadi, 2016, 2). Deteksjonen gir oss med andre ord også objekters koordinater i et bilde. Nettverket gir også betingede klassesannsynligheter for flere klasser. Disse regnes ut som sannsynligheten for klasse_I, gitt at et objekt er tilstede i boksen. Til slutt multipliseres de betingede klassesannsynlighetene med konfidensen til boksene. Dette gir nettverket klassespesifikke konfidenser som brukes til å bestemme et objekts klasse (Redmond et al., 2016, 2). Konfidensen forteller oss med andre ord både hvor sannsynlig det er at et spesifikt objekt er i den predikerte boksen, samt hvor god den predikerte boksen er.

Datagrunnlag

Makinlæringsalgoritmer krever store mengder data for å kunne optimere en generalisert modell for et problem. Innen klassifisering behøver vi data som representerer objekter i flest mulig orienteringer og omgivelser. Modellene kan kun generalisere for data som er en del av treningsgrunnlaget, noe som medfører at en modells robusthet og invarians, ofte direkte korrelerer med mengden variert data. Med dagens teknologi er det ofte ikke algoritmer og maskinvare som er den største begrensende faktoren innen maskinlæring, dette er derimot data.

Utover det å ha store mengder data, må også disse dataene annoteres, en prosess som er meget viktig innen denne typen maskinlæring, supervised learning. Dette er måten vi forteller nettverket hva en batch med data er, og hvor objektene befinner seg i bildet. Uten annoteringsfiler er dataene meningsløse. Dette fordi nettverket da ikke vet hva dataene representerer, altså vet ikke nettverket hva det skal gjenkjenne i dataene. Annoteringsfiler inneholder informasjon om

objektets klasse, representert i figur 8 som første tall i hver linje, og posisjon i bildet, som er gitt ved de fire siste tallene.

Dataaugmentering er en metode som kan brukes til å utvide treningsgrunnlaget, når dette ikke er tilstrekkelig stort eller mangfoldig. Prosessen innebærer enkle databehandlingsteknikker for å produsere syntetisk data fra eksisterende data, og resulterer i en større datamengde samtidig som en ivaretar variasjon i data (Ng, 2017d). Innenfor maskinsyn er vanlige dataaugmenterings-teknikker eksempelvis speiling, tilfeldige utklipp, fargeendringer og flere andre. Darknet, og andre maskinlæringsbiblioteker implementerer flere former for augmentering som automatisk gjennomføres under trening. To av disse er i Darknet en funksjon for tilfeldig opp- og nedskaling av bilder, samt speiling av bilder.

```
1 0.716797 0.395833 0.216406 0.147222
0 0.687109 0.379167 0.255469 0.158333
1 0.420312 0.395833 0.140625 0.166667
```

Figur 8 – Eksempel på Darknet annoteringsfil

2.5 Trening

Før nettverket kan brukes til å klassifisere data, må nettverket trenes. Videre vil vi definere og forklare noen begreper som er sentrale i treningen av nevrale konvolusjonsnettverk.

Loss-funksjon

Loss-funksjonen, også kalt errorfunksjonen, er en funksjon som beskriver hvor mye et nettverks estimerte output er feilkalkulert i forhold til den forventede output. For å beregne dette estimatet bruker nettverket de aktuelle vektene og bias for å avgjøre hva et bilde fra datasettet representerer. Funksjonen bruker så dette estimatet til å sammenligne estimatet med forventet output, som vi har gjennom annoteringsfiler. Under trening av et konvolusjonsnettverk er målet å optimere cost-funksjonen, som er summen av loss, mot dens minimale verdi. Dette oppnås ved å finjustere vektene steg for steg via gradient descent.

Gradient descent

Gradient descent er en metode for å optimere ”cost”-funksjonen til et nevralgt nettverk ned mot funksjonens minima. Cost-funksjonen, annotert som J i formel 2, er summen av ”loss”, annotert som L , over hele datasettet. Funksjonen har parametrene w = vekter, b = biaser og m = datasampler i datasettet. $\hat{y}(i)$ er nettverkets estimerte output gitt input i , og $y(i)$ er korrekt output gitt i annoteringsfilen.

$$J(w, b) = \frac{1}{m} \sum_i^m L(\hat{y}(i, w, b), y(i)) \quad (2)$$

Gradient descent søker å optimere cost-funksjonen ved å finne de optimale verdiene for w og b gjennom metoden vist i formel 3 og 4.

$$w = w_0 - \alpha \frac{\partial J(w, b)}{\partial w} \quad (3)$$

$$b = b_0 - \alpha \frac{\partial J(w, b)}{\partial b} \quad (4)$$

I formlene 3 og 4, representerer α læreraten i nettverket. I praksis vil vi ikke minimere cost-funksjonen til absolutte minimum, men vi ønsker den så lav som mulig uten at en kommer inn i overfitting-området. Denne prosessen kan i dype konvolusjonsnettverk ta veldig lang tid, om en starter med tilfeldige vekter, med andre ord om en trener fra ”scratch”. En metode som minimerer denne lange prosessenes kalles ”transfer learning”. Denne metoden innebærer å starte trening med vekter som allerede er trent på et annet datasett. Disse vektene vil allerede være tilpasset til å gjenkjenne mange typiske karakteristikker som vil være like på tvers av datasett. Eksempler på disse karakteristikkene er enkle karakteristikker som sirkler, kanter og andre enkle former. Egenskapen ved å gjenkjenne disse formene blir så overført til de nye vektene en trener, og kan dramatisk minimeres tiden det tar å trenere et nevralgt nettverk (Ng, 2017c).

Forward og backpropagation

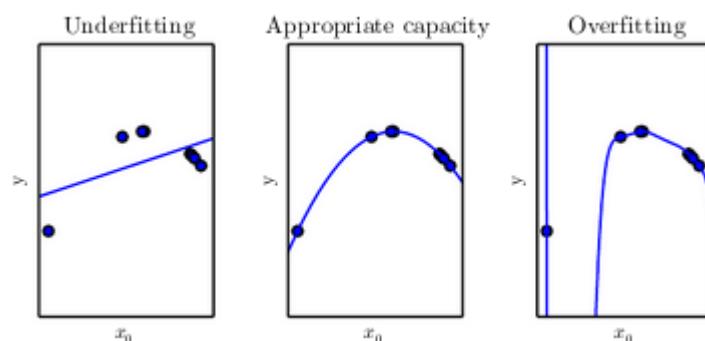
”Propagation” i et nevralgt nettverk kan i sin enkleste form anses som dataflyt gjennom nettverkets forskjellige lag. Når nettverket får en input x , for eksempel et bilde med sine piksler, flyter denne dataen gjennom nettverket. Dette produserer en output \hat{y} , som gir et estimat på hvorvidt det er et objekt i bildet og hvilken klasse objektet tilhører. Estimatelet kan videre brukes til å regne ut loss for en batch med data. Dette kalles ”forward propagation”. Når vi trener et nettverk vil data også flyte bakover i nettverket. Dette kalles ”backpropagation” og brukes for å regne ut gradienten til cost-funksjonen, fra formel 3 og 4, som er essensiell informasjon for å legge til rette for nettverkets læring (Goodfellow et al., 2016, 200).

Lærerate

Læreraten, α fra formel 3 og 4, er et desimaltall som nettverket bruker i gradient descent for å vite hvor mye vektene skal justeres for hver treningsiterasjon. Læreraten kan sees på som hvor store steg nettverket tar når det lærer (Goodfellow et al., 2016. 83). Dersom læreraten er for stor risikerer en å ikke treffe de mest optimale verdiene for vektene. Ved å derimot velge for liten lærerate vil en oppleve at optimeringen tar lang tid.

Over-/underfitting

Når vi trener et nevralgt nettverk har vi i utgangspunktet to datasett, ett treningssett og ett valideringssett. Vi trener nettverket på treningssettet, og det er på disse dataene at vi optimerer cost-funksjonen. I hovedsak kan en få tre mulige utfall, eller slutttilstander, på vektene våre som vist under i figur 9.



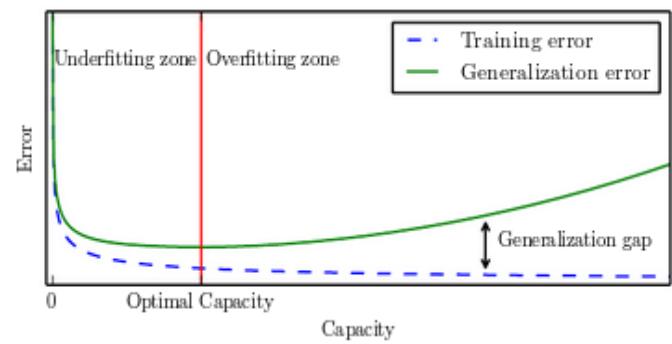
Figur 9 - Forskjellig tilpassede modeller

(Goodfellow et al., 2016, 112)

Det første ekstreme tilfellet er det som kalles «underfitting». I dette tilfellet er ikke vektene tilpasset treningssettet godt nok, som vil gi store utslag i cost-funksjonen. Ved denne tilstanden klarer ikke nettverket å gjøre pålitelige deteksjoner på treningsdataene, og er da sannsynligvis ikke pålitelig nok på annen data heller.

Det andre ekstreme tilfellet kalles «overfitting», som er det tilfelle vi får når vektene er for godt tilpasset treningsdataene. Nettverket vil klare å gi oss veldig gode deteksjoner på treningsdataene, men vil ikke kunne generalisere like godt på data som ikke er en del av treningsgrunnlaget.

Området som det er ønskelig å ligge i, «optimal capacity», er i praksis det området hvor «generaliseringsgapet» er minst (Goodfellow et al., 2016, 111). Generaliseringsgapet er differansen mellom trenings- og generaliseringsfeil. Med treningsfeil menes loss på treningssettet, mens generaliseringsfeil er loss på valideringssettet, altså data som ikke er en del av treningsgrunnlaget. I figur 10 visualiseres generaliseringsgapet i forhold til underfitting, overfitting og optimal capacity.



Figur 10 – Generaliseringsgap (Goodfellow et al., 2016, 113)

Mean Average Precision

Mean Average Precision (mAP) er modellens gjennomsnittspresisjon over alle klasser i datasettet (Shah, 2018). Tallet, et prosentmål på hvor godt en ferdig trent modell klarer å klassifisere objekter i test-datasettet, er en verdi som vi bestemmer etter at treningen er ferdig. Presisjon er definert i formel 5, og mAP er gjennomsnittet av presisjonen for alle klasser i datasettet.

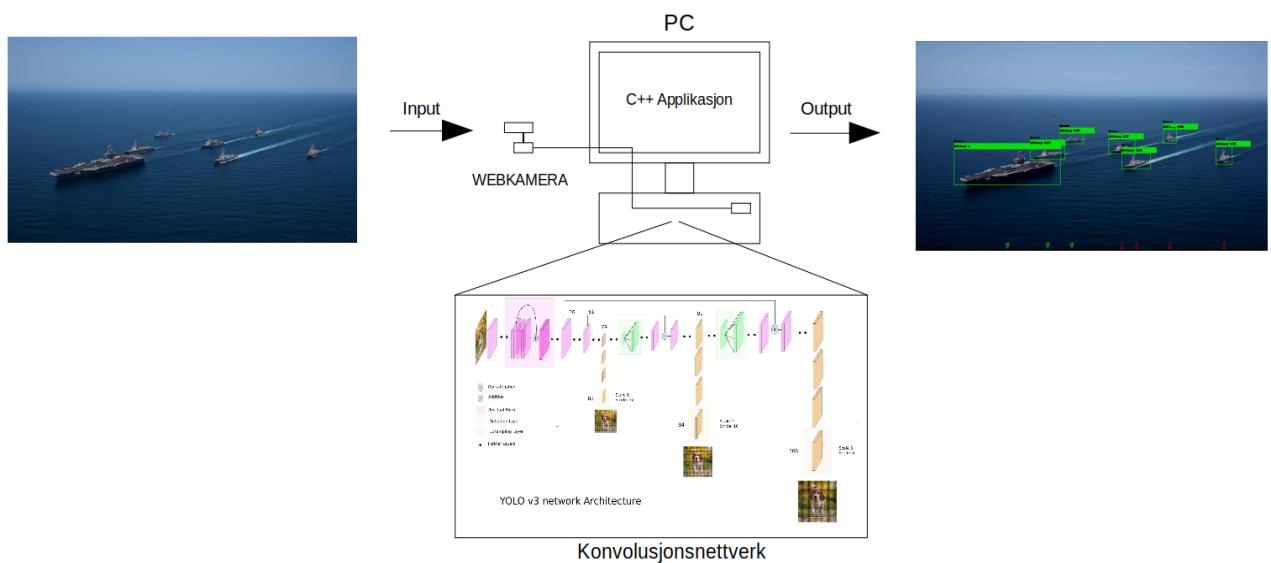
$$TP = \text{Ekte positiver}$$

$$FP = \text{Falske Positiver}$$

$$\text{Presisjon} = \frac{TP}{TP + FP} \quad (5)$$

3 Systemdesign

Kapittel tre tar for seg utarbeidelsen av gjenkjenningssystemet. Her presenteres valg av maskinvare og programvare, samt konfigurasjonene og programmeringen som har vært nødvendig for å lage produktet. Kapittelet er delt inn i fem delkapitler; maskinvare, programvare, konfigurasjon av YOLOv3, treningsgrunnlag og validering og bruk. Alle egenproduserte programkoder, samt en liste over programmene som er åpen-kildekode finnes i vedlegg H.



Figur 11 - Systemtegning

I figur 11 kan en se en enkel systemtegning av klassifiseringssystemet. Systemet er bygget opp med en input som er et bilde fanget opp med webkamera fra operasjonsområdet. Videre prosesseres bilderammen og analyseres av nettverket. Til slutt får bruker ut en output på skjermen som viser tydelige bokser rundt de klassifiserte objektene. Disse boksene inneholder klassevis konfidens og relativ kurs. Boksene er på formatet midtpunkt, bredde og høyde, som vi visualiserer i vår C++-applikasjon. I tillegg til boksen vil brukeren få opp en pil nede på skjermen som viser objektets relative peiling.

3.1 Maskinvare

Maskinvaren vi har brukt i denne oppgaven er hovedsakelig et produkt som ble bygget i forbindelse med en tidligere oppgave innen maskinlæring. Komponentene som har blitt brukt er beskrevet i tabell 3. På bakgrunn av kravet om prosesseringshastighet var vi nødt til å akseletere bildeprosesseringen ved å bruke Compute Unified Device Architecture (CUDA), som igjen medførte at vi måtte bruke GPUer fra leverandøren NVIDIA i bildeprosessering. For å kunne bruke CUDA, var vi nødt til å ha GPUer med en compute capability, et mål på en GPUs generelle spesifikasjoner, som var høyere enn 3,0. Vi valgte å bruke to GTX 1080Ti til dette formålet, begge med compute capability lik 6,1.

KOMPONENT	BESKRIVELSE
RANDOM ACCESS MEMORY (RAM)	3 * 8GB Kingston ValueRAM DDR4
PROSESSOR	Intel Pentium G4400 @ 3.3GHz * 2
GPU	2 * GTX 1080Ti 11GB NVIDIA
DISK	Samsung 960 EVO 250 GB SSD
OPERATIVSYSTEM (OS)	Ubuntu 16.04 LTS

Tabell 3 - Maskinvare

3.2 Programvare

I arbeidet med denne oppgaven har vi anvendt flere forskjellige åpen-kildekode hjelpe midler. Det viktigste av disse er rammeverket som har blitt brukt til maskinlæring, Darknet. For at Darknet skulle kunne trenne nettverket vårt ved hjelp av en GPU ble også programvarene CUDA og CUDA Deep Neural Network (cuDNN), samt OpenCV installert. Sistnevnte program har vært nødvendig for å kunne behandle video fra webkameraet vårt.

Valg av programvare ble gjort på bakgrunn av de krav vi analyserte som essensielle for det ferdige systemet. Kravet som ble prioritert var kravet til konvolusjonsnettverkets proseseringstid av hensyn til bildebehandling. En minimering av prosesseringstiden anså vi som kritisk for at informasjonen, levert av nettverket, videre skulle kunne anvendes til å ta hurtige avgjørelser i en operativ setting.

For å velge nettverksarkitektur, og som en følge av dette også rammeverk, samlet vi inn informasjon om prestasjonen til noen av de eksisterende løsningene med høyest prestasjon. Tallene under er hentet fra en artikkel om YOLOv3 (Redmond & Farhadi, 2018, 1). De forskjellige modellene er testet på datasettet COCO, som er et populært datasett for å teste modellers prestasjon.

MODEL	MAP	FPS
SSD500	46.5	19
YOLOV3-608	57.9	20
RETINANET 50 500	50.9	14
SSD513	50.4	8
FAST RCNN	59.1	6

Tabell 4 - Ulike modellers prestasjon etter testing

Tallene i tabell 4, er bakgrunnen for at vi valgte å benytte oss av arkitekturen YOLOv3. Arkitekturen gjør det mulig å analysere video i sanntid med den maskinvaren vi har hatt tilgjengelig for oppgaven.

Med dette valget, var det nødvendig å bruke rammeverket Darknet. Programvaren er skrevet i programmeringsspråket C, et lavnivå språk som kan utføres meget hurtig på datamaskiner.

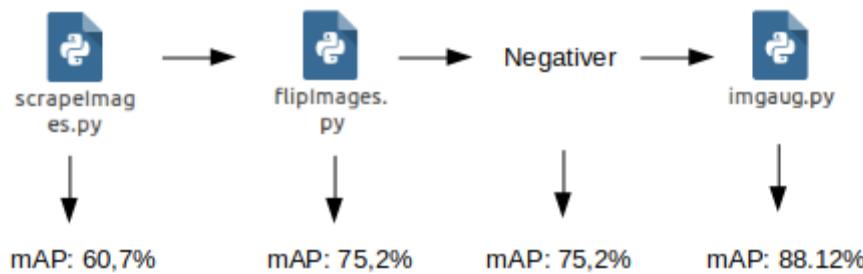
3.3 Konfigurasjon av YOLOv3

For å bruke nettverket til klassifisering av både fartøystype og relativ kurs, var vi nødt til å definere seks klasser. De seks klassene vi definerte var *civilian_right*, *civilian_left*, *civilian_parallel*, *military_right*, *military_left* og *military_parallel*. For å kunne klassifisere disse seks klassene var vi nødt til å gjøre noen endringer i nettverkets konfigurasjonsfil. Disse endringene ble gjort i nettverkets tre deteksjonslag, hvor vi satt: *classes* = 6 og *filters* = 33, satt i henhold til regelen: *filters* = (*classes* + 5)*3, i siste konvolusjonslag før deteksjon (AlexeyAB, 2018).

Videre har vi også skrudd av en data-augmenteringsfunksjon som speiler treningsdataene mens nettverket trenes ved å sette *flip* = 0. Dette er gjort på grunn av at vi ønsker at fartøyets relative kurs skal være en del av deteksjonen som nettverket gjør. Videre har vi etter testing besluttet å trenne modellene med konfigurasjonen *random* = 1. Dette medfører at vi opp- og nedskalerer nettverksoppløsningen i inputlaget underveis i treningsprosessen. Denne skalingen skal i teorien resultere i en modell som er bedre rustet til å detektere objekter i flere forskjellige skalaer. Videre fører denne augmenteringsfunksjonen til at datasettet effektivt blir større som en følge av at systemet har trent med forskjellig oppløsning på alle bildene.

3.4 Treningsgrunnlag

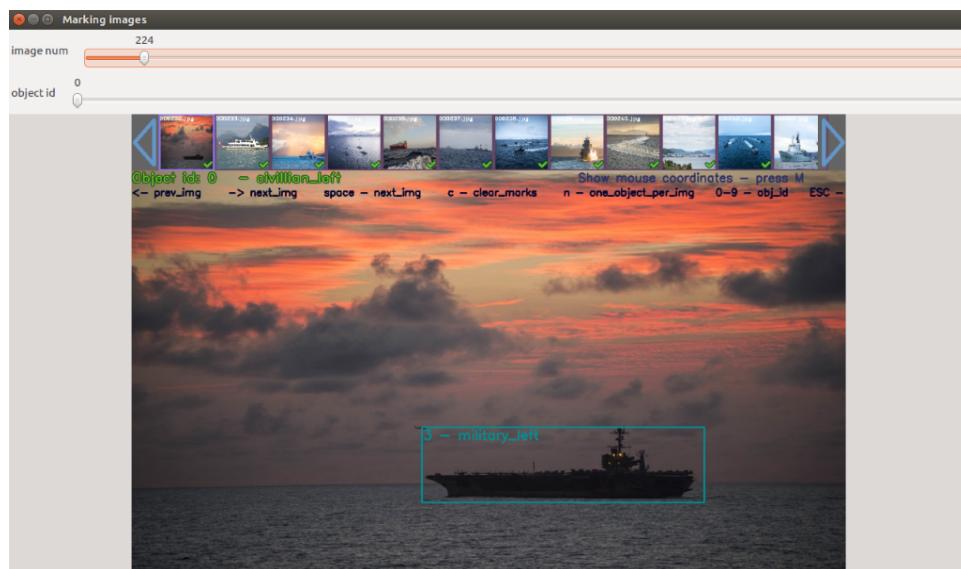
Vårt endelige datasett er totalt bestående av 7643 bilder med tilhørende annoteringsfiler. Av disse er 7043 bilder til trening og de resterende 600 bildene brukes til validering av modellen. Å ha et godt og variert datagrunnlag har vært et viktig fokusområde i arbeidet med denne oppgaven. Det faktum at vi har produsert datasettet vårt selv har gjort det til et viktig aspekt som potensielt kunne lagt krav på store deler av den tiden vi har hatt tilgjengelig til å arbeide med oppgaven. Vi har hatt en ambisjon om å minimere denne tidsbruken, som har resultert i den valgte fremgangsmetoden for å produsere vårt datasett.



Figur 12 - Prosessen ved produsering av datasett

Produksjonen av datasettet har vi, som tidligere nevnt, delt inn i fire hovedfaser; web-scraping, augmentering ved speiling, negative bilder, samt augmentering ved cropping.

Flere av annoteringsfilene er produsert gjennom bruk av et åpen-kildekode verktøy, kalt *Yolo_mark*, til å annotere bildene i datasettet. Resterende annoteringsfiler er produsert automatisk gjennom egenproduserte script.



Figur 13 - Skjerm bilde fra annoteringsprosessen

Fase 1: Web-scraping

Grunnstammen i datasettet vårt er produsert ved å bruke et åpen-kildekode script, *scrapeImages.py*. Scriptet samler inn et gitt antall bilder som vi spesifiserer ved en søkestreng, og lagrer bildene til en spesifisert filbane. Grunnet mye dårlig data filtrerte vi bort rundt 50% av dataene samlet inn med denne metoden. Videre har vi også dratt ut i skjærgården rundt Bergen og

fotografert fartøy selv, men vi erfarte raskt at dette var en ineffektiv metode for å samle data i det størrelsesomfanget vi trengte for å trenne en robust nok modell. Etter filtrering av bildene satt vi igjen med om lag 3200 bilder, hvorav 600 bilder har blitt brukt til validering som etter manuell annotering gjorde at vi ble klare for å trenne vår første modell.

Fase 2: Augmentering ved speiling

Vi supplerte deretter datasettet med bilder av objekter som modellen oppnådde lavest presisjon på, nemlig bilder av fartøy med relativ kurs mot høyre og venstre. Dette gjorde vi med det egenutviklede scriptet, *flipImages.py*. Datasettet besto etter denne fasen av nærmere 3800 bilder. Siden de nye dataene var et produkt av data-augmentering med grunnlag i treningssettet ble ingen av de nye bildene lagt til i valideringssettet. Dette ble gjort for å unngå falskt høye presisjonsresultater etter validering.

Fase 3: Negative bilder

I den hensikt å gjøre modellen mer invariant til bildenes bakgrunn la vi til 800 negativer, altså “tomme” bilder uten objekter. Bildene til dette er hentet ved hjelp av web-scraping og tomme annoteringsfiler til disse bildene er generert ved å bruke *rename.py*.

Fase 4: Augmentering ved cropping

For å ha data i store nok kvantum til å kunne produsere en robust modell utviklet vi scriptet *imgaug.py*. Scriptet tar hele datasettet som input og klipper ut $\frac{3}{4}$ av bildene fra et tilfeldig startpunkt, og lagrer utklippene som nye bilder. Den store fordelen med å bruke dette scriptet, er tidsbesparelsen vi oppnår ved at scriptet automatisk annoterer de nye bildene. Denne løsningen doblett datagrunnlaget vårt, og produserte det som har blitt det endelige datasettet.

Utover å forbedre vår modell ved å endre treningsgrunnlaget har vi også utforsket aspektet med læreratestyring. Tidlig i arbeidet med oppgaven testet vi forskjellige typer læreratestyring, hovedsakelig læreratestyring styrt av hvor langt vi har kommet i treningsprosessen, og syklistisk læreratestyring. Den stegvise fremgangsmåten bruker en høyere lærerate i de første fasene av trening for å hurtigere tvinge loss-funksjonen tilnærmet sitt optimale område. Denne fremgangsmåten er vanlig å bruke, og gav oss tidlig gode resultater. Vi forsøkte også å implementere syklistisk læreratestyring ved å skrive *CLR.py* og bruke disse verdiene til lærerate-

styring (Smith, 2017, 4). Vår implementering er ikke lik den beskrevet i rapporten av Smith, men bruker noen av de samme prinsippene. I praksis innebærer denne typen læreratesyring at vi syklisk bytter mellom et høyt og et lavt nivå på læreraten, hvor gjennomsnittet av disse verdiene synker eksponentielt. Vi opplevde dårligere resultater ved bruk av denne metoden, og har holdt oss til den mer tradisjonelle stegvise læreratestyringen gjennom resten av arbeidet.

For å minimere tiden vi brukte på å trenere nettverket, startet vi all trening med vekter som var forhåndstrent på et annet datasett. Disse vektene er gjort tilgjengelig av arkitektene bak Darknet. Denne metoden kalles transfer learning, og muliggjør at vi kan utnytte noen av disse vektenes egenskaper som fører til at modellen ikke behøver å trenere for å gjenkjenne enkle former selv (Ng, 2017c).

Vedlegg A viser resultater fra treningsperioden etter hver fase av dataproduksjon.

3.5 Validering og bruk

For å teste modellene etter trening, utviklet vi *plot_mAP.py*. Dette scriptet bruker funksjonalitet innebygd i Darknet til å teste presisjonen nettverket oppnår på validering- eller treningssettet. I løpet av en treningsfase blir vektene lagret for hvert hundrede steg, og scriptet vi har utviklet for å validere modellen, tester for presisjon ved å bruke samtlige av disse vektfilene. Vi har valgt å bruke mAP til å vurdere resultatene fra trening.

Vi har bygget en C++-applikasjon som bruker Darknet som et ”shared-library” for å gjøre deteksjoner på bilderammer fra et webkamera ved hjelp av OpenCV. Applikasjonen laster opp vår modifiserte nettverksarkitektur, samt våre ferdigtrente vekter, og bruker dette til å klassifisere bilderammer fra videoen. Grunnen til at vi har laget en applikasjon, fremfor å bare kjøre modellen gjennom terminalen, er at det muliggjør bedre kontroll over presentasjonen av nettverkets deteksjoner. Vi henter ut deteksjon på klasse, koordinater og presisjon, og presenterer dataene på et format som mennesker hurtig kan forstå visuelt. Dette inkluderer fartøystype, relativ kurs og relativ peiling. Et eksempel på dette vises i figur 11.

4 Resultater

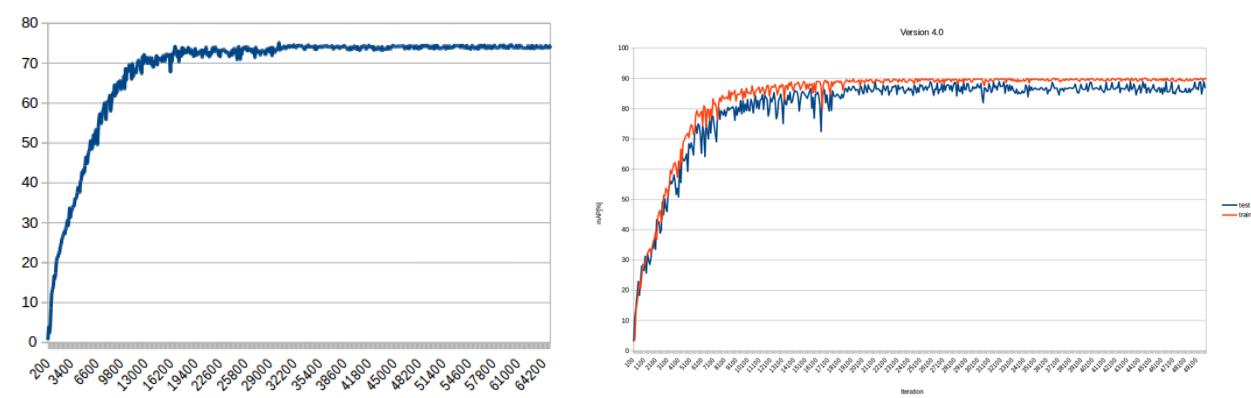
Delkapittelet vil presentere resultatene fra tester av klassifiseringssystemets konfidens og prestasjon gjort underveis i utviklingen av systemet. I tillegg fremvises tilbakemeldingene fra gjennomført demonstrasjon av klassifiseringssystemet, samt resultater fra påfølgende intervju.

4.1 Testing av klassifiseringssystemet

I denne delen vil vi presentere resultater oppnådd gjennom tekniske tester av systemets prestasjon på ulike områder. Først vil resultatene fra presisjonstesting av systemet ved ulike stadier av utviklingen bli presentert. Disse resultatene er presentert i mer utfyllende detalj i vedlegg A. Videre presenterer vi resultater fra tester av systemet i ulike miljø.

Presisjonstesting

Gjennom utviklingen av systemet har vektene, som kan ansees som systemets overførte læring eksistert i forskjellige tilstander. Disse vektene, og de forskjellige verdiene de har hatt gjennom utviklingen, resulterer i forskjellig prestasjon og presisjon når anvendt av nettverket. Presisjonsmålet mAP er et mål på hvor stor prosentandel av nettverkets klassifiseringer er korrekte når brukt på et valideringssett. All presisjonstesting har blitt gjennomført ved bruk av *plot_mAP.py*. Dette verktøyet har hjulpet oss med å teste presisjonen nettverket har oppnådd ved en gitt treningstilstand, og plottet disse resultatene grafisk. Grafene i figur 14 illustrerer hvordan resultatene ser ut, og er relativt like ved alle stadiene av utvikling, sett bort fra verdiene. Vi viser derfor ikke alle grafene i dette avsnittet, men samtlige er lagt frem i vedlegg A.



Figur 14 - mAP valideringsdata (fase 3) og validerings- og treningsdata (fase 4)

Figur 14 viser data fra plot_mAP.py, og visualiserer presisjonen vektene har oppnådd i henholdsvis fase tre og fire. I grafen til venstre kan en se at presisjonen på valideringssettet stabiliserer seg rundt 75%. Videre oppnådde de beste vektene fra fase tre 91.3% presisjon da de ble testet mot treningsdataene. Grafen til høyre visualiserer generaliseringsgapet etter fase fire. Her vises både mAP mot valideringsdataene i blått, og treningsdataene i oransje. Disse verdiene stabiliserer seg på rundt henholdsvis 88% og like under 90%. Disse tallene viser at vi har klart å gjøre generaliseringsgapet tilstrekkelig lite.

De fire ulike utviklingsstadiene av treningsgrunnlaget har hver sin tilhørende trenings- og testingsperiode. I tabell 5 presenteres oppnådd gjennomsnittlig presisjon på valideringsdataene oppgitt for hvert stadium av denne utviklingen.

FASE	MAP	GENERALISERINGSGAP
1	60,7%	-
2	75,2%	-
3	75,2%	Tilnærmet 16%
4	88,1%	Tilnærmet 1,5%

Tabell 5 - Utvikling av presisjon

Prestasjon gjennom periskop

Ved demonstrasjon av systemet, avholdt på Ubåtsenteret på Haakonsvern Orlogsstasjon, fikk vi tilgang og tillatelse til å teste systemet på diverse videoer, fanget gjennom periskopet på en Ula-klasse ubåt. Observasjoner fra denne testingen er at modellen detekterer opp mot 100% konfidens på de delene av videoene med høy kvalitet. Konfidensen i deteksjonene avtar i takt med synkende kvalitet på videoen, helt frem til konfidensen faller under 50%, og systemet forkaster deteksjonen. Videre detekterer systemet i noen tilfeller noen av høydestrekene i periskopet som fartøy på horisonten når disse er parallelle. Det sistnevnte tilfellet er vist i bildet oppe til høyre i figur 15, der systemet har en deteksjon på høydestreken med 64% konfidens.



Figur 15 - Test gjennom periskop. Her med konfidens 100%, 64% og 96%

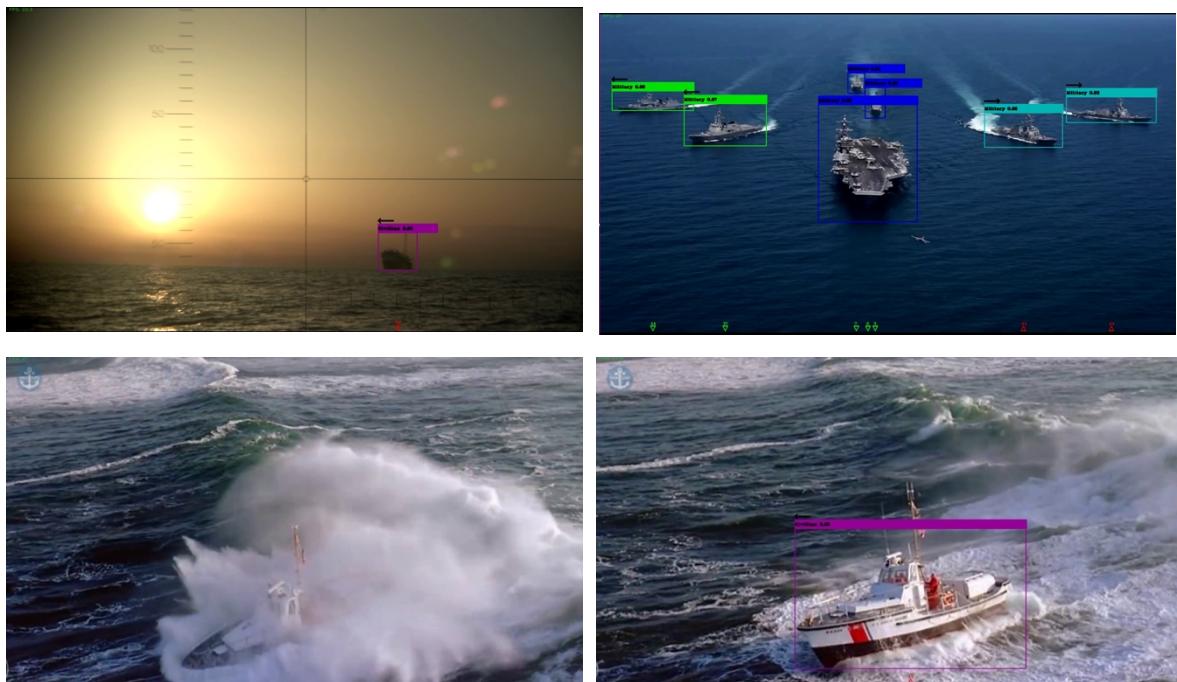
Prestasjon i varierende lys- og værforhold

Tester i nattforhold produserte ingen deteksjoner. Videre har vi observert at konfidensen i deteksjoner på bilder i dårlige lysforhold synker tilnærmet proporsjonalt med at mengden lys i bildet avtar. Den samme effekten observeres på bilder med utfordrende værforhold, som for eksempel en storm. I det øyeblikket fartøyene blir mer tydelige, og ikke er gjemt av sjø og vær, gjør nettverket deteksjoner som er tilfredsstillende. I noen tilfeller har vi også observert

at vanskelige forhold resulterer i ukorrekte deteksjoner på klasse med høy nok konfidens til at deteksjonen visualiseres.

Ved tester under optimale forhold har vi observert meget gode resultater. Klassifiseringene har vært nesten utelukkende av meget høy konfidens, med svært få ukorrekte klassifiseringer. Videre er det også såpass få bilderammer hvor objekter ikke klassifiseres, at sporingen av fartøy observeres som meget stabil. Når vi sier optimale forhold, refereres det til at forholdene er optimale i forhold til at de er meget like vær- og lysforholdene i en stor andel av treningsgrunnlaget. Ved noen tilfeller har vi observert at hangarskip feilklassifiseres som sivile fartøy, men relativ kurs klassifiseres korrekt.

Videoer fra disse testene er vedlagt på minnepenn under vedlegg G.



Figur 16 - Dårlige lysforhold (oppe venstre), optimale forhold (oppe høyre) og høy sjø (to nederste bilder)

Bildene i figur 16 eksemplifiserer systemets varierende prestasjon under forskjellige forhold. I bildet oppe til venstre gir systemet deteksjon med konfidens på 83%, og i bildet oppe til høyre har det en konfidens fra 83% til 98%. Nede til venstre kan en tydelig se at systemet sliter ved mye sjø, og konfidensen er her under terskelen for klassifisering, som er på 50%. Det siste bildet, nede til høyre, har en deteksjon med konfidens på 62%.

4.2 Demonstrasjon og intervju

I dette kapittelet vil tilbakemeldinger fra demonstrasjon og besvarelser fra påfølgende intervju bli presentert. Hensikten med demonstrasjonen og intervjuet var å avdekke i hvilken grad et klassifiseringssystem kan være et bidrag inn i Marinens Krigføringssenter. På demonstrasjonen var det representanter primært fra Ubåttjenesten, 1. Fregattskvadron og Marinens Krigføringssenter.

Intervjuspørsmålene avdekket flere områder hvor et maskinlæringsssystem, som tar for seg deteksjon og klassifisering, kan være nyttig. Vi har valgt å presentere de områdene som har vist seg å by på flest utfordringer per dags dato, og hvor det tilsynelatende kan se ut til at nye systemer, som vårt klassifiseringssystem, kan implementeres for å øke operativ evne.

Svarene fra intervjuene kan utslettes på forespørsel og er ikke lagt ved i vedleggene. Dette fordi svarene har blitt sendt på gradert forma og kan inneholde gradert informasjon.

Menneskelige begrensinger

Det er flere områder ved mennesket som er en begrensning i operativ sammenheng. Vi har etter å ha lest svarene fra intervjuobjektene valgt å se på menneskets utfordringer innenfor syn og utholdenhetsbegrensning. Intervjuobjektene beskriver menneskeøyet som en viktig sensor for å danne overflatebilde, og inputen herfra er med på å danne beslutningsgrunnlag for videre handling. Videre har mennesket begrenset utholdenhetsbegrensning, noe som påvirker fokus og beslutningstakingen. Det er stor enighet blant de ansatte som har blitt intervjuet, at systemer som hjelper til med gjenkjenning bidrar til avlastning hos operatøren om bord.

Tap av kvalifisert personell

”Visuell gjenkjenning hos utkikk krever kompetanse og erfaring” skriver et av intervjuobjektene som svarte på e-postintervjuet. Dette er skrevet som en forlengelse av at en av utfordringene Marinens Krigføringssenter står ovenfor er gjennomtrekk av personell, og dermed tap av operativ kompetanse. Intervjuobjektet sier videre at tap av kompetent personell i størst grad gjelder når det kommer til funksjoner som kles av vernepliktige. Her ser en ofte at innen vedkommende har opparbeidet seg nok kompetanse er det tid for dimisjon, og fartøyet står ovenfor et nytt opplæringsbehov. Videre har det kommet frem av intervjuene at maskinlæringsssystemet vil kunne være et faglig hjelpemiddel for operatøren som videre rapporterer om fartøyer av interesse.

Deteksjon og klassifisering under vanskelige forhold

Marinen operer ikke bare i dagslys og under gode forhold. I de fleste tilfeller skal en kunne skille mellom ”venn eller fiende” i nattemørket, ved mye sjø eller i hurtig skiftende omgivelser. Flere av intervjuobjektene påpeker at det er ved lange avstander og dårlige lys- og værforhold at menneskelige operatører kunne trengt et støtteverktøy i utførelse av klassifisering. Det er under slike forhold at beslutningstaking om bord går tregere, nettopp fordi det er vanskeligere å prosessere informasjon når det er vanskeligere å klassifisere objekter på havoverflaten.

Database

Forsvaret har gode ressurser og forutsetninger for å drive datainnsamling. Det vi har erfart etter demonstrasjon og diskusjon med ansatte er at det mangler et system for innsamling av data. Noe data samles på, men denne dataen blir sjeldent delt mellom grener eller avdelinger.

5 Drøfting

Oppgavens drøftingsdel er delt i to underkapitler. Den første delen tar for seg drøfting rundt resultatene fra tester av klassifiseringssystemet. Den andre delen av drøftingen diskuterer hvordan et klassifiseringssystem kan påvirke operativ evne om bord Marinens fartøyer.

5.1 Klassifiseringssystemet

Drøftingen i denne delen tar hovedsakelig utgangspunkt i de testene gjort gjennom utviklingen av modellen. Her tar vi for oss diskusjon rundt presisjonstesting, bruk av modellen med periskop og modellens kapasitet under varierende forhold. I tillegg til dette har vi også drøftet de tekniske valgene vi gjorde ved utarbeidelse av systemet.

Som nevnt tidligere i oppgaven ønsket vi at vårt system skulle evne å produsere informasjon en kunne bruke for å ta operasjonelle avgjørelser. Med dette som en ønsket slutttilstand ble prosesseringstid en høyst viktig faktor under prosessen med å velge nettverksarkitektur og rammeverk. Som vist i tabell 3, eksisterer løsninger som kan levere mer presise deteksjoner enn valgt arkitektur. Ulempen med samtlige av disse løsningene er at vi mister egenskapen med å kunne prosessere video i sanntid. I situasjoner hvor sanntids dataprosessering ikke er kritisk, vil det muligens være gunstig å gå for en mer regneoperasjonstung løsning, som kan produsere enda sikrere deteksjoner enn vår valgte løsning. Med fokus på behandlingstid satt til en så stor viktighet, som gjort i denne oppgaven, er YOLOv3 den utvilsomt beste løsningen når en veier presisjon opp mot behandlingstid.

Valget om å lage et enkelt klassifiseringssystem bestående av kun seks klasser har ført til at vi nådde målet vårt om å vise at et slikt system potensielt kan ta over eller avlaste oppgaver som i dag gjøres av menneskelige operatører. De seks klassene har ført til at vi kan skille mellom militære og sivile fartøyer, samt ta ut relativ kurs. Dersom et slikt system skulle vært implementert i Marinens ville det vært behov for en økning av antall klasser. Dette fordi det er behov for å kunne klassifisere ulike klasser av marinefartøy og sivile fartøy. Med slik informasjon vil operatørene om bord kunne brukt klassifiseringsverktøyet som en sensor for å danne

et mer detaljert overflatebilde. En økning av antall klasser vil kreve store utvidelser av datasettet.

Presisjonstesting

Vi har lagt stort fokus på dataaugmentering under produksjonen av oppgavens datasett. Vi ser tydelig av resultatene av presisjonstesting, at større tilskudd av variert data øker systemets presisjon betraktelig. Denne presisjonen bør også kunne økes ytterligere med et enda større treningsgrunnlag, men en kan ikke forvente en lineær økning i presisjon som en funksjon av mer data. Det er også diskuterbart om tilskuddet av negative bilder hadde noe bidrag til modellens prestasjon. Vi kan av prestasjonstesting ikke si med sikkerhet at tilskuddet av disse dataene har vært et konstruktivt bidrag, da vi ikke oppnådde noen nevneverdige økninger i presisjon. Det er dog tenkelig at modellen har blitt mer invariant til objektenes bakgrunn, uten at vi kan bevise dette med den metoden for testing vi har valgt.

Under utviklingsfase en og to har det ikke blitt gjennomført noen presisjonstesting på treningsdataene, men kun på valideringsdata. Dette ble utelatt på bakgrunn av manglende konfidens på at generaliseringsgapet ved disse tidlige stadiene av utvikling var små nok til at modellen ikke var overtrent. Disse antakelsene blir bekreftet av testingen i fase tre. Presisjonen, som kommer som et resultat av testing i fase fire, viser at systemets presisjon på treningssettet har avtatt. Denne utviklingen kommer tydelig frem i figur 14. Dette kan være, og er mest sannsynlig, et resultat av at treningssettet i sin totalitet er vanskeligere å optimere på som en følge av tilskuddene fra dataaugmentering-scriptet, *imgaug.py*. Dette presisjonsfallet er et akseptabelt tap som kommer av at presisjonen på valideringssettet har økt i det størrelsesomfanget vi har observert, og at generaliseringsgapet har blitt helt nede i 1,5%. Etter fase fire oppnår systemet en presisjon på over 88% mot valideringsdataene, et resultat vi er meget fornøyde med, og som er godt over kravene vi stilte til systemet. Videre er generaliseringsgapet blitt såpass lite at vi føler oss trygge på at vi har en godt tilpasset modell, som ikke er overtrent.

Periskop

Resultatene fra denne testen er bedre enn forventet, med tanke på vinkelhastigheten til kameraet som har fanget opp videoen. Så lenge bildene er relativt like treningsgrunnlaget produserer systemet mange gode klassifiseringer, som potensielt kan bidra til å øke situasjonsforståelsen.

sen om bord en ubåt. Likevel er det større deler av videoene hvor systemet presterer dårlig. De varierende resultatene i deteksjonene på videoene gjennom periskop kan forklares av flere årsaker. Den primære årsaken til det som i flere tilfeller kan ansees som dårlige resultater, er videoenes kvalitet. Videoene er fanget av et kamera som filmer gjennom et optisk periskop og kvaliteten på videoene er ikke optimale. Det faktum at de avsnittene med høy kvalitet produserer deteksjoner med høy konfidens, og de avsnittene med lavere kvalitet resulterer i mindre sikre deteksjoner understøtter dette. En videre faktor som bidrar til de observerte resultatene kan tenkes å være trådkorset og de andre detaljene i videoen som nettverket aldri har sett i treningsgrunnlaget. Dette eksemplifiseres ved at høydestrekene i noen tilfeller detekteres som fartøy helt i horisonten. En potensiell løsning på disse effektene er å trene nettverket, der bildene i treningsgrunnlaget har dette trådkorset lagt over alle bildene.

Varierende forhold

Systemets prestasjon under dårlige vær- og lysforhold var forventet å være relativt dårlige. Dette er en konsekvens av at andelen bilder med nettopp dårlige forhold i treningsgrunnlaget er underrepresentert i forhold til andelen bilder i gode forhold. Treningseksemplarer i nattemørke er intensjonelt ekskludert som en del av oppgavebegrensningen og mangelen av Red-Green-Blue-bildeinformasjon (RGB) i nattbilder. Når denne begrensningen er gjort i produksjonen av treningsgrunnlaget, er det teknisk ikke mulig å detektere fartøy om natten. En mulig løsning for å detektere fartøy om natten kan være å bygge opp et datasett med bilder av fartøy gjennom et infrarødt kamera (IR-kamera), da denne typen kamera kan hente ut større mengder informasjon om objekter om natten. Av den samme årsaken som at fartøy i nattemørket ikke detekteres, presterer systemet dårlig på fartøy i storm og andre utfordrende værforhold. Disse treningseksemplarene eksisterer ikke, og modellen har dermed ingen grunnlag for å detektere dem. Selv i uvær er en andel av bildeammene i en video relativt like treningsgrunnlaget, noe som resulterer i deteksjoner. Disse deteksjonene er dog ikke like sikre som de er ved optimale forhold, men dette kan skyldes at bildene, som en effekt av forstyrrelsene fra uværet, blir av dårligere kvalitet.

Testen på video i optimale forhold produserer meget gode resultater. Klassifiseringene er korrekte, har en gjennomgående meget høy konfidens, og gir stabil sporing av fartøy. Denne testens resultater eksemplifiserer hvordan et system for fartøysklassifisering og målangivelse kan fungere, om bord fartøy i Marinens. Selv om systemet ikke er feilfritt, da noen hangarskip blir feilklassifisert i enkelte tilfeller, viser testen tydelig teknologiens potensiale. De gode resulta-

tene fra denne testen skyldes det faktum at kildevideoen er av høy kvalitet, stabil, samt at lys- og værforhold er relativt likt som systemets treningsgrunnlag. Om et liknende system blir produsert profesjonelt, vil treningsgrunnlaget være mye bredere enn det vi har produsert i denne oppgaven. Det er sannsynlig at resultater sett i denne testen vil kunne oppleves for alle forhold systemet utvikles for å håndtere.

Returverdier

Systemet returnerer ikke utelukkende klasse, relativ kurs og peiling, men også posisjon. For hver deteksjon systemet gjør, returnerer nettverket koordinatene til detekterte objekters midtpunkt til datamaskinen. Disse koordinatene forteller maskinen nøyaktig hvor i bildet et objekt er posisjonert. Om vi vet i hvilken sektor rundt et marinefartøy et bilde som sendes inn i nettverket representerer, samt gjør en avstandsberegning, vil systemet kunne si med høy nøyaktighet akkurat hvor et fartøy befinner seg relativt til eget fartøy. Denne typen informasjon kan brukes til å automatisk gjennomføre målangivelser, som kan videresendes til andre systemer om bord. Det skal dog nevnes at avstandsberegning ikke er noe denne oppgaven har tatt for seg, og enten må gjøres uavhengig av vårt system, eller må implementeres.

5.2 Maskinlæring som klassifiseringssystem

Drøftingen i dette delkapittelet baserer seg i stor grad på funnene gjort ved demonstrasjon av klassifiseringssystemet og gjennom påfølgende intervju. De samme kategoriene som ble presentert i resultatkapittelet er bakgrunnen for diskusjonen.

Menneskelige begrensinger

Som nevnt i resultatkapittelet er det sikreste visuelle deteksjonsverktøyet vi har å forholde oss til på fartøy i dag, menneskeøyet. Vi benytter oss i dag av flere hjelpe midler, som eksempelvis høyt plasserte kameraer, for å assistere oss i overvåkningen av gitt ansvarsområde. Til tross for slike hjelpe midler er det mennesker som analyserer og vurderer overflatebildet. Det vi ser per i dag er at vår beste ressurs på området deteksjon og klassifisering visuelt er begrenset til ”line of sight” og dermed horisonten på en fin dag. I tillegg til synsrekkevidde er også synsfelt en begrensning hos mennesket. Vi klarer ikke analysere det vi ikke ser, og vi klarer kun å koncentrere oss om et mindre område om gangen.

Et kamera vil også være begrenset til horisonten, men til forskjell fra menneskeøyet har et kamera mulighet til å bruke optisk zoom for å gi oss høyoppløselige bilder av fjernobjekter. Dette er en kapabilitet menneskeøyet ikke kan konkurrere mot. I tillegg vil et automatisert system, likende det vi har laget, kunne brukes med flere kameraer som har 360 graders dekning. En kan dermed analysere bilder fra hele ansvarsområdet på flere skjermer samtidig. Videre vil en maskin ikke bli påvirket av utmattelse og dermed miste fokus etter lengre tid med høyere belastning, slik som mennesker vil.

Til tross for at en maskin i flere tilfeller vil ha høyere kapasitet mangler et maskinlæringssystem resonnement. Vårt system baserer seg på øyeblinksvurderinger og gir dermed deteksjoner og klassifisering som er absolutt. Dette kan medføre at systemet ignorerer viktig informasjon som er knyttet til sammenhengen mellom forskjellige øyeblinksbilder, og ikke ser den store sammenhengen. Et eksempel på dette er at feilaktig deteksjon av relativ kurs kan forbli lik, selv om en over flere bilder ser at relativ kurs er en annen enn klassifisert. Videre mangler vårt system flere sanser som dagens operatører har, for eksempel hørsel. Informasjon som kan oppfattes gjennom denne sansen blir ikke en del av klassifiseringsprosessen, med mindre det

implementeres. På den andre siden er det ikke sikkert at et slikt system trenger resonnement om det kun skal brukes til beslutningsstøtte. Det kan holde at systemet tar seg av klassifiseringen, og viser dette tydelig i et grafisk brukergrensesnitt, som overlater resonnement og beslutningstaking til operatøren.

Tap av kvalifisert personell

Om bord er det stadig utskifting av personell, og som oftest er det vernepliktig personell som byttes ut. Vernepliktige nyttet i funksjoner som eksempelvis utkikk, en av funksjonene om bord som detekterer og klassifiserer fartøy. En utfordring flere ansatte i Marinens har påpekt er at vernepliktig personell byttes ut når de omsider har fått god kompetanse og er på et godt kvalifisert nivå når det kommer til deteksjon og klassifisering. En kan derfor argumentere for at ved å implementere en datamaskin som bidrar til å utføre den samme jobben, som for eksempelvis en utkikk på bro, vil ikke besetningen lide av at kunnskap og kompetanse forsvinner på dette området. En datamaskin dimitterer ikke og dersom treningsgrunnlaget utvides vil prestasjonen øke så lenge systemet er i drift. Videre vil ikke et slikt system kun være en permanent del av mannskapet og sikre en viktig funksjon, men det vil også kunne avlaste personellet om bord ved å automatisere noen av funksjonene som i dag gjøres av mennesker.

Et klassifiseringssystem er et godt verktøy innenfor ”decision support systems”.² Ved å benytte seg av dette systemet vil en også kunne anta at effektiviteten, når det kommer til kommunikasjon, øker. Dette fordi overflatebildet vil kunne deles med flere operatører samtidig da bildet kan vises på flere skjermer samtidig. Til tross for de nevnte argumentene for å implementere et slikt system, kan en fallgruve være avhengigheten et slikt system kan skape blant personellet om bord. En tenkt konsekvens av å ha en maskin som kommer med deteksjonsalternativer for operatørene kan være latskap. Dette kan videre føre til at en neglisjerer å verifisere de klassifiseringer systemet gjør automatisk. Dette kan videre føre til flere etiske dilemmaer som vi har valgt å ikke diskutere i denne oppgaven.

² Beslutningsstøtteverktøy

Deteksjon og klassifisering under vanskelige forhold

Mennesket ser mye når forholdene legger til rette for det, men når utfordrende vær, mørke eller hurtig skiftende omgivelser er det som omgir oss, blir det vanskeligere å detektere og klassifisere objekter. Når bilder blir uklare og en i tillegg på ett forsøk skal kunne se om dette er vennlig eller fiendtlig fartøy kan det bli vanskelig for selv det best trente øyet. En løsning for å øke mulighet for deteksjon og klassifisering i nattemørke, kan være å kombinere maskinlæring med IR-kamera. Det skal også nevnes at datamaskiner tolker bilder som matriser av intensitetsverdier. Bilder i dårlige lysforhold inneholder ofte svake nyanseforskjeller som er vanskelige for mennesker å se. Et klassifiseringssystem kan derfor i slike tilfeller lese disse forskjellene bedre enn mennesker. Videre kan vårt system prosessere 22 til 25 bilder i sekundet. Maskinvareoppdateringer vil kunne øke dette antallet. Dette medfører at systemet vil kunne klassifisere bilder hurtigere enn mennesket, samt analysere bilder av høyere kvalitet, noe som vil være gunstig ved for eksempel hurtig sveip med periskop.

Vanskelige forhold er per i dag også en utfordring for systemet vi har laget. Utfordrende forhold, som ikke likner på treningsgrunnlaget, medfører feildeteksjon og i mange tilfeller klarer ikke systemet å detektere objekter i det hele tatt. I teorien skal dette kunne bedres ved å inkludere et større spekter av bilder med utfordrende forhold i treningsgrunnlaget. Microsoft Research team beviste i 2015 at maskinlæringsmodeller, som har hatt nok data tilgjengelig for trening, oppnådde bedre prestasjoner enn godt trente mennesker innen bildeklassifisering (He et al., 2015).

Database

En database bestående av ulike fartøy tatt i ulike vær- og lysforhold er det behov for. En plattform for deling av informasjon på tvers av Marinens, og Forsvarets, avdelinger vil kunne være med på å øke den totale kompetansen. Forsvaret har gode ressurser og forutsetninger for å drive datainnsamling og annotering. Per i dag kan en utfordring med å implementere et klassifiseringssystem være det faktum at vi ikke har et stort nok datagrunnlag til å gjøre systemet pålitelig og robust. Til tross for dette kan systemet bidra til datainnsamling og ferdig annoterte bilder ved å lagre øyeblikksbilder, men dette krever at systemet har et godt treningsgrunnlag til å begynne med. Dersom Marinens påbegynner slik datainnsamling, kan dette legge til rette for utviklingen av robuste klassifiseringssystemer. Dette har vi fått belyst gjennom tester av systemet vi har laget for oppgaven.

En betrakning som vi ikke har undersøkt i denne oppgaven er den faktiske muligheten for implementering av et klassifiseringssystem. Faktorer som her vil være avgjørende er kvalifisert personell til utvikling, kostnader knyttet til utvikling og implementering, gradering av systemet og anskaffelser av hardware. I dag er det vanlig at Forsvaret enten kjøper et ferdig system fra industrien, eller leier inn konsulenter som lager et system. En utfordring med et klassifiseringssystem er at det aldri blir ”ferdig”. Dette fordi det stadig vil dukke opp nye fartøy som ikke er klassifisert og lagt inn i systemets treningsgrunnlag. En mulighet kan være å etablere en egen avdeling som jobber med å trenere og utvikle klassifiseringssystemene.

6 Konklusjon med anbefaling

Med en sikkerhetspolitisk situasjon som stadig endrer seg; en endring i trusselbilde og et krav om høyere beredskap, har det blitt et større fokus på økt stridsevne i Marinens Sammen med langtidsplanen er det vedtatt at det skal innføres autonome og ubemannede systemer som skal minimere risiko og arbeidsbelastning, samt øke situasjonsforståelsen. Det skrives i langtidsplanen at ”det skal satses videre på informasjons- og kommunikasjonsteknologi (IKT) i forsvarssektoren for å tilrettelegge for at Forsvaret skal kunne løse sine viktigste oppgaver, og for å bidra til god utnyttelse av sektorens ressurser” (Forsvarsdepartementet, 2015, 8). Gjennom denne oppgaven har vi produsert og presentert et system innenfor maskinlæring som i stor grad kan anvendes til å automatisere, overta eller avlaste oppgaver som i dag gjøres av operatører i Marinens Videre kan et klassifiseringssystem være et bidrag inn i en høyteknologisk marine med tanke på bildebygging, beslutningstaking, avlastning av personell og bevaring av kompetanse.

Gjennom intervjuene har vi avdekket at det vil være behjelplig med et system som kan bidra til hurtigere og en mer sikker visuell deteksjon og klassifisering. Dette for å øke beslutningsgrunnlaget, samt effektivisere beslutningsprosessen. Om et slikt system bygges til å anvende IR-kamera og optisk zoom kan systemet levere pålitelig informasjon i situasjoner hvor nåtidens ressurs, mennesket, har prestasjonsmessige utfordringer. Eksempler på disse situasjoner er tilfeller hvor objekter er skjult av mørke eller i horisonten.

En annen utfordring som et maskinlæringssystem kan lette på er tap av kvalifisert personell og overbelastning hos mannskapet om bord. Et slikt system som vi har utviklet til denne bacheloroppgaven vil være en løsning for å beholde kompetanse nettopp fordi en datamaskin ikke dimiterer. Videre vil systemet kunne avlaste personellet om bord ved å gi klassifiseringsalternativer ved deteksjon, samtidig som at maskinen ikke påvirkes av faktorer som lite søvn og har derfor en utholdenhetsnivå vi mennesker ikke innehar.

De tekniske resultatene vi har oppnådd i denne oppgaven mener vi viser at teknologien innenfor maskinlæring er modent for å bli et satningsområde i Forsvaret. Systemet vi har laget produserer gode resultater innenfor oppgavebegrensningene med relativt høy konfidens og

effektivitet. Når vi ser på systemets prestasjon på bilder vi har utelatt fra treningsgrunnlaget ser vi et markant fall i deteksjonsprestasjon som viser noen av manglene vårt system har. Dette eksemplifiserer det faktum at store og varierte datamengder er essensielt innen maskinlæring.

Til tross for gode argumenter for at et maskinlæringsssystem bør satses på i en ny og høyteknologisk marine, er det noen utfordringer som må tas i betraktning. Et slikt autonomt system har ikke evnen til å resonnere, og har per nå ikke andre inputs enn kameraet. Et menneske i samme funksjon har flere sanser, samt mulighet til å vurdere omgivelsene på bakgrunn av flere faktorer. Videre kan det tenke seg at et autonomt system gjør at mannskapet tar snarveier og neglisjerer verifiseringene av systemets klassifisering.

På bakgrunn av oppgavens tema og de resultatene vi har oppnådd, konkluderer vi med at maskinlæring som teknologi er modent for operativ bruk og er et fagområde som Forsvaret bør satse på. Videre har våre tekniske tester bekreftet viktigheten av datainnsamling, og vi anbefaler derfor at Forsvaret i første omgang implementerer prosedyrer for datainnsamling og annotering til bruk til maskinlæringsformål.

Bibliografi

- AlexeyAB (2018). Darknet. Hentet fra <https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects> 16.06.18
- Bradski, G. & Kaehler, A. (2008). *Learning OpenCV*. USA: O'Reilly Media Inc.
- Dettmers, T. (2018). *Deep Learning in a nutshell: Core concepts*. Nvidia Devblogs. Hentet fra <https://devblogs.nvidia.com/deep-learning-nutshell-core-concepts/> 14.06.18
- Forsvarsdepartementet (2015). *Kampkraft og bærekraft. Langtidsplanen for forsvarssektoren*. (Prop. 151 S 2015-2016) Hentet fra <https://www.regjeringen.no/contentassets/a712fb233b2542af8df07e2628b3386d/no/pdfs/prp201520160151000dddpdfs.pdf> 26.06.18
- Geigety, A. (2014). *Machine Learning Is Fun: Part 4*. Medium. Hentet fra <https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471> 27.05.18
- Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. Massachusetts: MIT Press.
- Hardesty, L. (2017). *Explained: Neural Networks*. Hentet fra <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>. 16.08.18
- Hareide, O.S., Relling, T., Pettersen, A., Sauter, A., Mjelde, F.V., & Ostnes, R. (2018). Fremtidens autonome ubemannede kapasiteter i Sjøforsvaret. I R. Espevik (Red.), *NECESSE* (Vol.3, 2.utg., 123-148). Bergen: Sjøkrigsskolen
- He, K., Zhang, X., Ren, S. & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. Hentet fra <https://arxiv.org/pdf/1502.01852v1.pdf%20%20.pdf> 16.11.18
- Huang, T.S. (1996). *Computer Vision: Evolution and Promise*. Hentet fra <https://cds.cern.ch/record/400313/files/p21.pdf> 26.06.18
- Hui, J. (2018). *Real-time Object Detection With YOLO, YOLOv2 and now YOLOv3*. Medium. Hentet fra https://medium.com/@jonathan_hui/real-time-object-detection-with-yolov2-28b1b93e2088 25.06.18

-
- Ng, A. [Deeplearning.ai] (2017a, 7. November). *C4W1L02 Edge Detection Examples* [videofil]. Hentet fra
<https://www.youtube.com/watch?v=XuD4C8vJzEQ&index=2&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF>
- Ng, A. [Deeplearning.ai] (2017b, 7. November). *C4W2L03 Resnets* [videofil]. Hentet fra
<https://www.youtube.com/watch?v=ZILibUvp5lk&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF&index=14>
- Ng, A. [Deeplearning.ai] (2017c, 7. November). *C4W2L09 Transfer Learning* [videofil]. Hentet fra
<https://www.youtube.com/watch?v=FQM13HkEfBk&index=20&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF>
- Ng, A. [Deeplearning.ai] (2017d, 7. November). *C4W2L10 Data Augmentation* [videofil]. Hentet fra
<https://www.youtube.com/watch?v=JI8saFjK84o&index=21&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF>
- Nielsen, M. (2018). Kapittel 1. *Neural Networks and Deep Learning*. Determination Press.
Hentet fra <http://neuralnetworksanddeeplearning.com/chap1.html>
- Kathuria, A. (2018). *Whats New in YOLOv3?*. Towards Data Science. Hentet fra
<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b> 26.06.18
- Redmond, J. & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. Hentet fra
<https://pjreddie.com/media/files/papers/YOLOv3.pdf> 25.06.18
- Redmond, J., Divvala, S., Girshik, R. & Farhadi, A. (2016). *You Only Look Once: Unified Real-time object detection*. https://pjreddie.com/media/files/papers/yolo_1.pdf 26.06.18
- Sagdahl, M. (2017). *Autonomi*. Hentet fra <https://snl.no/autonomi> 10.10.18
- Shah, T. (2018). *Measuring Object Detection Models – mAP – What is Mean Average Precision?*. Towards Data Science. Hentet fra <https://towardsdatascience.com/what-is-map-understanding-the-statistic-of-choice-for-comparing-object-detection-models-1ea4f67a9dbd> 25.06.18
- Sheridan, T.B & Verplank, W.L. (1978). *Human and computer control of undersea teleoperators*. Hentet fra <https://apps.dtic.mil/dtic/tr/fulltext/u2/a057655.pdf> 01.11.18
- Smith, L.N. (2017). *Cyclical Learning Rates For Training Neural Networks*. Hentet fra
<https://arxiv.org/pdf/1506.01186.pdf> 27.06.18

Zeiler, M. & Fergus, R. (2013). *Visualizing and Understanding Convolutional Networks*.

Hentet fra <https://arxiv.org/pdf/1311.2901.pdf> 25.06.18

Vedlegg

Vedleggene D, E, F og G er av praktiske årsaker ikke lagt ved papirutgaven av denne oppgaven. Disse vedleggene ligger på minnepenn levert sammen med oppgaven, og kan også bli gjort tilgjengelig til enkeltpersoner ved å ta kontakt med forfatterne av bacheloroppgaven.

Vedlegg A – Trening

A.1 - Trening 1

Hensikten med denne første treningsrunden var todelt. Primært ønsket vi å teste hvorvidt darknet evner å detektere fartøyenes relative kurs, og som et sekundærmål å få en forståelse av de mengdene data vi behøver for å produsere en akseptabel modell. Vi anså det som uhensiktsmessig å bruke tid på å validere mot treningssettet på dette tidspunktet, da valideringsgaps høyst sannsynlig ville være av en slik størrelsesorden, at modellen ikke vil produsere anvendbare deteksjoner.

Datasett:

DATASETT	STØRRELSE (CA.)
TRENING	2600
VALIDERING	600

Tabell 6 - Datasett trening

Læreratestyring:

Lærerate: 0.001

Antall batcher: 70000

Policy: Steps

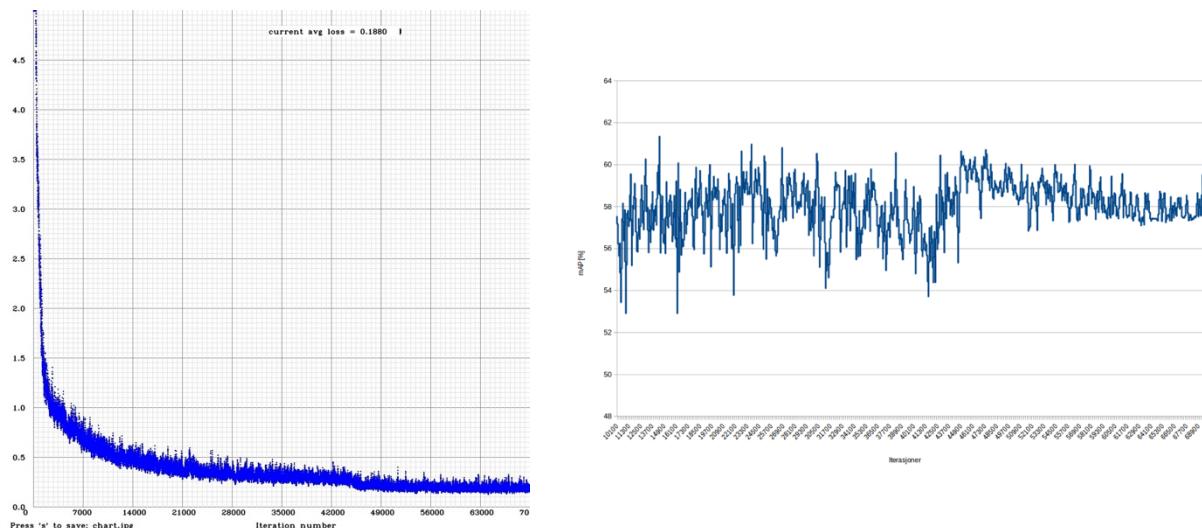
Steg: 30000, 45000

Skala: 0.1, 0.1

Denne typen læreratestyring endrer læreraten ved gitte intervaller i treningsprogresjonen. Den oppgitte læreraten over er verdien vi starter med ved initiering av treningen. Læreraten vil multipliseres med verdiene i "skala", ved de oppgitte "steg". Når nettverket har prosessert en batch med data, har den gjennomført et steg. I praksis fører denne tilnærmingen til læreratestyring at vi i den innledende fasen av treningen tar relativt store steg mot objektfunksjonens minima. Størrelsen på endringene i verdiene i vektene vi tar for hvert steg minkes når vi begynner å nærme oss minima, som en følge av at læreraten minkes med en faktor på ti. Med denne fremgangsmetoden skal vi i teorien optimere vektene uten å overskyte objektfunksjonens minima.

Vektene tilstand lagres ved hver hundrede steg, og disse vektene testes så mot valideringssettet for å måle presisjonen vi får på våre deteksjoner.

Resultater:



Figur 17 - Loss-chart til venstre og mAP til høyre

mAP: 60,7%

Dette tallet er den absolutt høyeste gjennomsnittlige presisjonen vi oppnådde etter å ha testet alle 7000 vektfiler som treningen resulterte i.

A.2 - Trening 2

Hensikten med denne runden var primært å undersøke resultatene av økninger i treningsgrunnlaget. En annen treningsparameter vi ønsket å undersøke resultatene av, var en innstilling vi gjør i hvordan bildene blir skalert på vei inn i nettverket. Setter vi: random lik null har alle bilder samme oppløsning på vei inn i nettverket. Endrer vi denne innstillingen til 1, vil darknet rotere hvilken oppløsning bildene har når de sendes inn i nettverket. I teorien skal dette kunne resultere i en modell som er bedre rustet til å detektere objekter av forskjellig størrelse. Vi ønsker å teste hvilken fremgangsmetode som resulterer i den beste modellen.

Datasett:

TYPE DATASETT	STØRRELSE (CA.)
TRENING	3200
VALIDERING	600

Tabell 7 - Datasett trening 2

Læreratestyring:

Lærerate: 0.001

Antall batcher: 65000

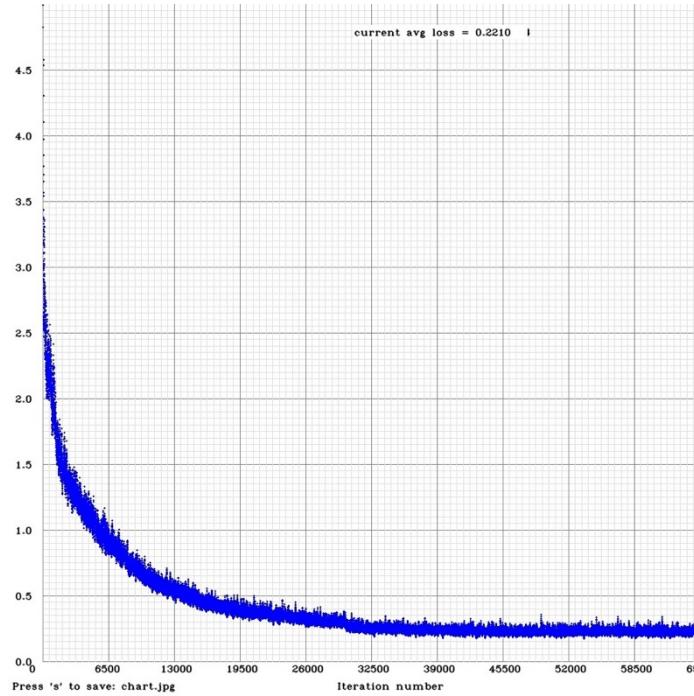
Policy: Steps

Steg: 25000, 45000

Skala: 0.1, 0.1

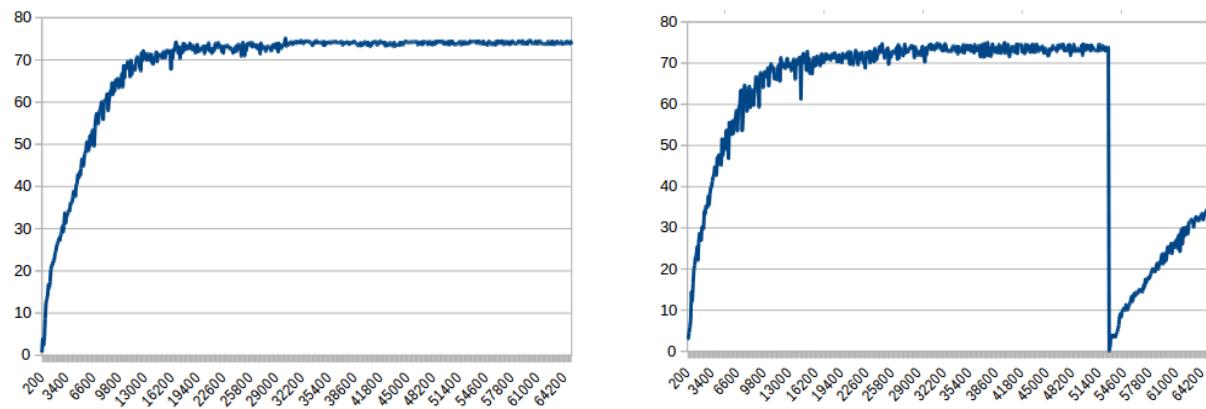
Resultater:

Loss-chart er tilnærmet lik for begge modeller. Av denne grunn viser vi kun den ene.



Figur 18 - Loss-chart, random = 0

Resultatene med hensyn på gjennomsnittlig presisjon var tilnærmet like, uavhengig av om vi trener med fast eller sirkulerende oppløsning. Årsaken til fallet i mAP for modellen med random lik en skyldes at vi gikk tom for lagringsplass under trening, og fortsette fra siste lagrede stadium.



Figur 19 - mAP med random = 0 til venstre og mAP med random = 1 til høyre

Uavhengig av hvilken modell vi valgte, lå presisjonen til de beste vektene på like rundt 75%.

mAP:75,2%

A.3 - Trening 3

Hensikten ved denne testen var å teste hvorvidt negative treningseksempler bedrer presisjonen modellen vår oppnår. I teorien skal trening med negative treningseksempler resultere i en modell som er mer invariant til objektenes bakgrunn. Et sekundærmål med treningen er å fastsette om vi får en bedre modell om vi videretrener våre tidligere trente vekter, eller om vi starter med initialiserte vekter fra darknet.

Datasett:

TYPE DATASETT	STØRRELSE (CA.)
TRENING	4000
VALIDERING	600

Tabell 8 - Datasett trening 3

Læreratesstyring:

Lærerate: 0.001

Antall batcher: 65000

Policy: Steps

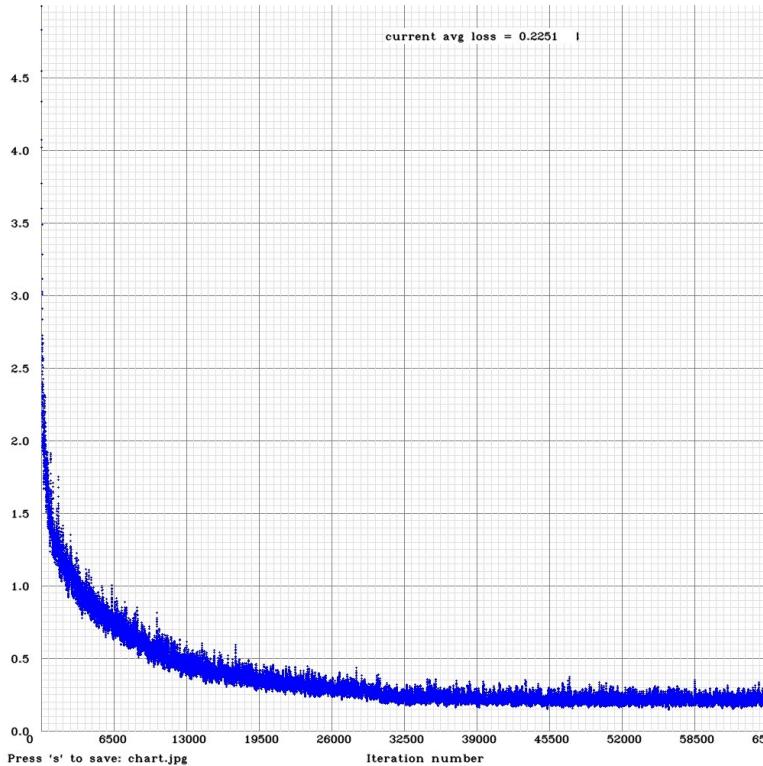
Steg: 25000, 45000

Skala: 0.1, 0.1

Random = 1

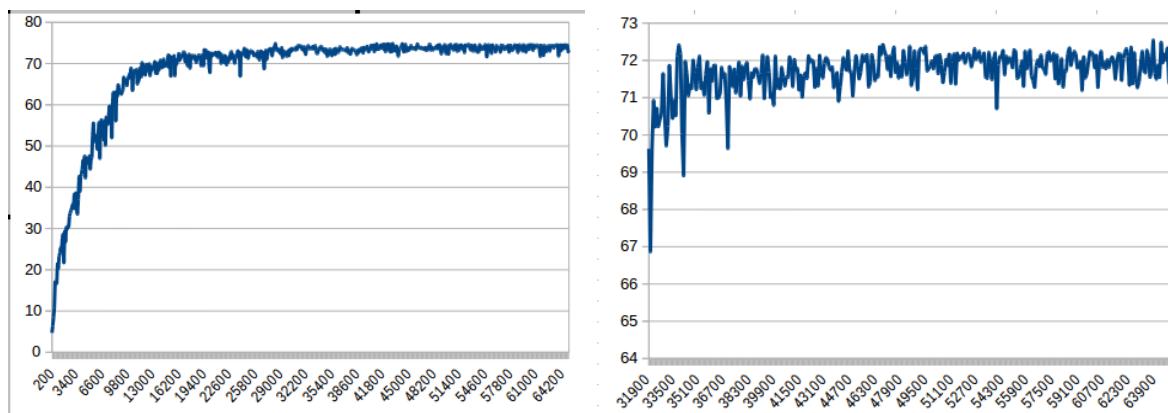
Resultater:

Loss-chart er fremdeles tilnærmet lik for begge treningsrunder.



Figur 20 - Loss-chart

Vi ser her en tydeligere forskjell på gjennomsnittlig presisjon, der modellen hvor vi videretterner vekter fra en tidligere modell kommer noe dårligere ut.



Figur 21 - mAP med initialiserte vekter til venstre og mAP med egne vekter til høyre

Modellens presisjon har ikke økt nevneverdig som en følge av å trenere med negativer, og vi har ingen måte å teste for modellens invarians til bakgrunn. De beste vektene oppnår en tilnærmet lik gjennomsnittlig presisjon mot valideringssettet.

mAP: 75,2%

Disse vektene oppnår en gjennomsnittlig presisjon på tilnærmet 92% mot treningssettet, som betyr at generaliseringsgapet fremdeles er stort.

```
detections_count = 7889, unique_truth_count = 5552
class_id = 0, name = civilian_left,      ap = 90.67 %
class_id = 1, name = civilian_right,     ap = 90.54 %
class_id = 2, name = civilian_parallel,   ap = 89.01 %
class_id = 3, name = military_left,       ap = 99.78 %
class_id = 4, name = military_right,      ap = 90.88 %
class_id = 5, name = military_parallel,    ap = 89.71 %
for thresh = 0.25, precision = 0.95, recall = 0.98, F1-score = 0.96
for thresh = 0.25, TP = 5448, FP = 311, FN = 104, average IoU = 80.07 %

mean average precision (mAP) = 0.917640, or 91.76 %
```

Figur 22 - Detaljerte mAP-data fra treningssett

A.4 - Trening 4

Etter betydelige økninger i treningsgrunnlaget, er hensikten å fastsette om vi nå har et treningsgrunnlag som er stort nok til å produsere en modell som presterer bra. Primærhensikten er å bestemme og visualisere generaliseringsgapet, samt å velge ut de best tilpassede vektene.

Datasett:

TYPE DATASETT	STØRRELSE
TRENING	7043
VALIDERING	600

Tabell 9 - Datasett trening 4

Læreratestyring:

Lærerate: 0.001

Antall batcher: 65000

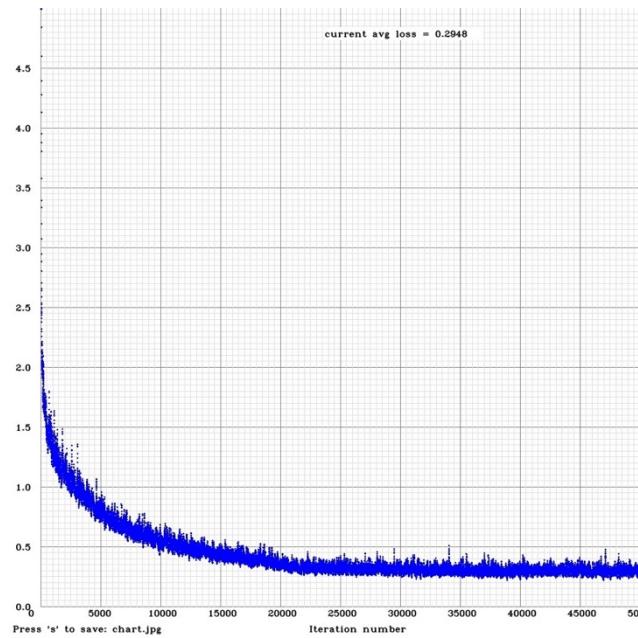
Policy: Steps

Steg: 25000, 45000

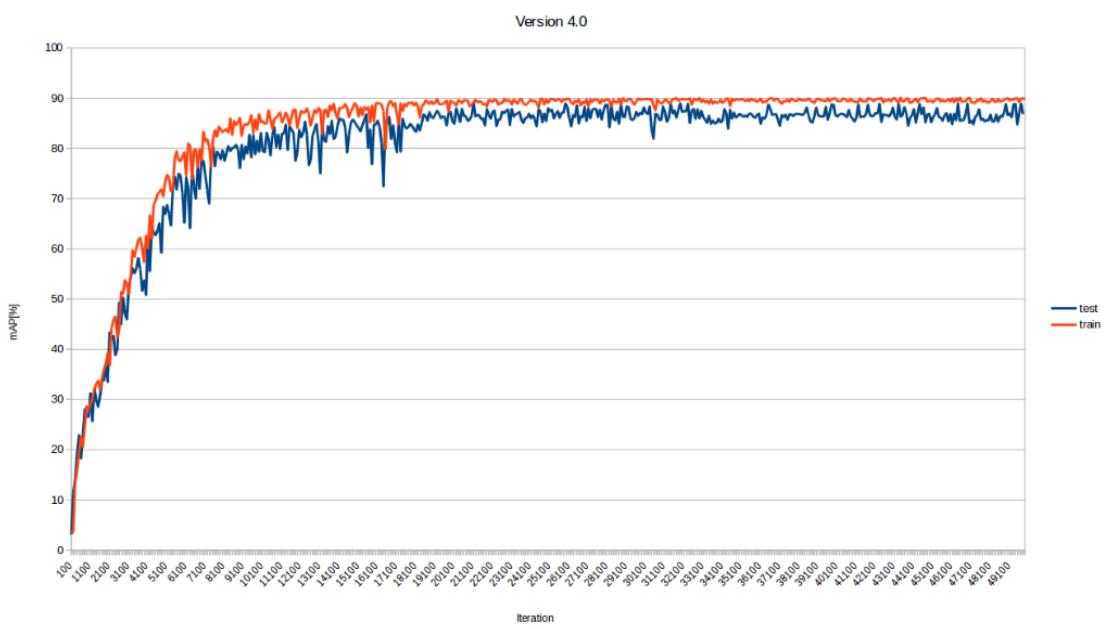
Skala: 0.1, 0.1

Random = 1

Resultater:



Figur 23 - Loss-chart



Figur 24 - mAP og generaliseringsgap

```

detections_count = 22184, unique_truth_count = 10962
class_id = 0, name = civilian_left, ap = 90.32 %
class_id = 1, name = civilian_right, ap = 90.12 %
class_id = 2, name = civilian_parallel, ap = 87.09 %
class_id = 3, name = military_left, ap = 90.00 %
class_id = 4, name = military_right, ap = 90.17 %
class_id = 5, name = military_parallel, ap = 89.06 %
for thresh = 0.25, precision = 0.92, recall = 0.95, F1-score = 0.94
for thresh = 0.25, TP = 10411, FP = 892, FN = 551, average IoU = 73.60 %

mean average precision (mAP) = 0.894593, or 89.46 %
Total Detection Time: 334.000000 Seconds

detections_count = 2424, unique_truth_count = 1181
class_id = 0, name = civilian_left, ap = 88.69 %
class_id = 1, name = civilian_right, ap = 89.23 %
class_id = 2, name = civilian_parallel, ap = 78.96 %
class_id = 3, name = military_left, ap = 95.14 %
class_id = 4, name = military_right, ap = 87.88 %
class_id = 5, name = military_parallel, ap = 88.92 %
for thresh = 0.25, precision = 0.90, recall = 0.98, F1-score = 0.90
for thresh = 0.25, TP = 1058, FP = 113, FN = 123, average IoU = 70.89 %

mean average precision (mAP) = 0.881237, or 88.12 %
Total Detection Time: 38.000000 Seconds

```

Figur 25 - Detaljert mAP for treningssett til venstre og for valideringssett til høyre

Presisjonen på treningssettet har avtatt noe etter at datasettet ble utvidet, men på grunn av at gjennomsnittspresisjonen på valideringssettet har steget betraktelig, ender vi opp med et generaliseringsgap på omtrent 1,5%.

mAP: 88,1%

Vedlegg B – Python Kildekode

B.1 – CLR.py

```
1 #CLR.py
2 #Python script for å produsere læreratestyring som forsøker å tilnerme sykliske lærerater
3 #Erlend Omland & Marthe Engvik
4 #I forbindelse med Bacheloroppgave i Militære studer med fordypning i Elektronikk og Data
5
6 steps = []
7 scales = []
8 step = 1000
9 i = 0
10 LR = 0.001
11 x = []
12 y = []
13 outFx = open("x.txt", "w")
14 outFy = open("y.txt", "w")
15
16 for i in range(70):
17     steps.append(step)
18     if i%2==0:
19         scales.append(0.91)
20         LR = LR*0.91
21         y.append(LR)
22         lr = str(LR)
23         outFy.write(lr)
24         outFy.write("\n")
25     else:
26         scales.append(1.03)
27         LR = LR*1.03
28         y.append(LR)
29         lr = str(LR)
30         outFy.write(lr)
31         outFy.write("\n")
32     x.append(i)
33     I = str(i)
34     outFx.write(I)
35     outFx.write("\n")
36     step = step + 1000
37
38 print(steps)
39
40 print(scales)
```

B.2 - create_train-test.py

```
1 #create_train-test.py
2 #Python script for å generere train.txt og test.txt, viktige filer for å kunne trenne modeller i darknet.
3 #Erlend Omland & Marthe Engvik
4 #I forbindelse med Bacheloroppgave i Militære studier med fordypning i Elektronikk og Data
5
6 import os
7
8 # Lager en liste av alle filer i en aktuell mappe, train/test
9 path = "/home/bachelor/data/DatasetDarknet/train"
10 dirs = os.listdir( path )
11 dirs.sort()
12
13 # Åpner en fil for å skrive filbaner for bildene som skal trenes/valideres på
14 # filnavn: train.txt/test.txt
15
16 outF = open("train.txt", "w")
17
18
19 # Skriver filbanen til alle .jpg bilder i aktuell mappe
20
21 for file in dirs:
22     filepath = "data/obj/"
23     filename = os.path.splitext(file)[0]
24     imgpath = filepath + filename + ".jpg"
25     outF.write(imgpath)
26     outF.write("\n")
27
28 outF.close()
```

B.3 - plot_mAP.py

```
1 #plot mAP.py
2 #Python script for å teste presisjon på validerings-/treningsdata, samt å generere filer til å plotte dataene
3 #Erlend Omland & Marthe Engvik
4 #I forbindelse med Bacheloroppgave i Militære studer med fordypning i Elektronikk og Data
5
6 import os, sys
7
8 # Lager en liste over alle vekt-filer i aktuell mappe
9 path = "/home/bachelor/darknet/backup"
10 weights = os.listdir( path )
11 weights.sort()
12
13 # Kjører presisjons-evaluering programvare, og skriver all output til fil "output.txt"
14 for weight in weights:
15     string = "./darknet detector map data/obj.data cfg/yolov3-Utkikk9000.cfg backup/"
16     string = string + weight + " -thresh 0.5"
17     string = string + " >> output.txt"
18     os.system(string)
19
20 # Oppretter filer for å skrive x- og y-koordinater til filer
21 outF = open("y.txt", "w")
22 outFi = open("x.txt", "w")
23
24 iteration = 100
25
26 # Leser relevant data fra "output.txt" og skriver data til filer
27 with open("output.txt", "r") as file:
28     for line in file:
29         if "mean average precision (mAP) =" in line:
30             it = str(iteration)
31             outFi.write(it)
32             outFi.write("\n")
33             outF.write(line[45:50])
34             outF.write("\n")
35             iteration = iteration + 100
36
37
38 outF.close()
```

B.4 - flipImages.py

```
1 #flipImages.py
2 #Python script for å augmentere treningsdataene. Scriptet speilier utvalgte bilder om en
3 #vertikal akse, og lagrer nye bilder med korrekte filnavn.
4 #Erlend Omland & Marthe Engvik
5 #I forbindelse med Bacheloroppgave i Militære studer med fordypning i Elektronikk og Data
6
7 import os
8 import cv2
9
10 path = "/home/bachelor/data/DatasetVOC/flip_theese"
11
12 # Lager liste over alle bildefiler som skal speiles
13 files = os.listdir(path)
14 files.sort()
15 first = 3349
16
17 # Speiler bilder, lagrer med nye filnavn
18 for file in files:
19     first += 1
20     name = "00"
21     name = name + str(first) + ".jpg"
22     image = cv2.imread(file)
23     cv2.flip(image, 1, image)
24     cv2.imwrite(name, image)
25     print(first)
26
27
```

B.5 - rename.py

```
1  # rename.py
2  # python script for å gi nye filnavn til negative datasamples, samt å generere annoteringsfiler
3  # Erlend Omland & Marthe Engvik
4  # I forbindelse med Bacheloroppgave i Militære studier med fordypning i Elektronikk og Data
5
6  import os
7
8  filepath = "/home/bachelor/data/Negatives/"
9
10 dirs = os.listdir(filepath)
11 dirs.sort()
12
13 first = 3751
14
15 # Gi nytt navn til filer
16 for dir in dirs:
17     path = "/home/bachelor/data/Negatives/" + dir
18     print(path)
19     ims = os.listdir(path)
20     for im in ims:
21         name = "00" + str(first) + ".jpg"
22         pic = path + "/" + im
23         os.rename(pic, name)
24         first = first + 1
25         textfile = "00" + str(first) + ".txt"
26         open(textfile, "w+")
27
```

B.6 - imgaug.py

```
1 # imgaug.py
2 # Python script for å augmentere treningsdata, samt å generere annoteringsfiler
3 # Erlend Omland & Marthe Engvik
4 # I forbindelse med Bacheloroppgave i Militære studier med fordypning i Elektronikk og Data
5
6 import cv2
7 import numpy as np
8 import os
9 import random
10
11 dirpath = "/home/bachelor/obj"
12 last = 4327
13
14 files = os.listdir(dirpath)
15 files.sort()
16
17 for file in files:
18     if file.endswith(".jpg"):
19         pic = cv2.imread(file)
20         W = 3 / 4 * np.size(pic, 1)
21         H = 3 / 4 * np.size(pic, 0)
22         w = int(W)
23         h = int(H)
24         seed_y = random.randint(0, ((1/4)*np.size(pic, 0)))
25         seed_x = random.randint(0, ((1/4)*np.size(pic, 1)))
26         # vi klipper bilder til 3/4 av bredde og høyde, med et random seed-punkt innenfor første 1/8 av h/b
27         crop = pic[seed_y:h+seed_y, seed_x:w+seed_x]
28         name = "00" + str(last) + ".jpg"
29         path = "/home/bachelor/cropped/" + name
30         # Lagrer croppet bilde
31         cv2.imwrite(path, crop)
32
33 tekstfil = os.path.splitext(file)[0] + ".txt"
34 File = open(tekstfil)
35 new = "00" + str(last) + ".txt"
36 Path = "/home/bachelor/cropped/" + new
37 Ofile = open(Path, 'w')
38 Pic = cv2.imread(file)
39 Cropfile = "/home/bachelor/cropped/00" + str(last) + ".jpg"
40 Cropped = cv2.imread(Cropfile)
41 for line in File:
42     fields = line.split(" ")
43     delete = True
44     obj_id = fields[0]
45     # størrelser i piksler, originalt bilde
46     center_x = float(fields[1]) * np.size(Pic, 1)
47     center_y = float(fields[2]) * np.size(Pic, 0)
48     width = float(fields[3]) * np.size(Pic, 1)
49     height = float(fields[4]) * np.size(Pic, 0)
50     # alle størrelser nå ganget med gammel/ny størrelse + seed
51     ny_x = (center_x - seed_x) / np.size(Cropped, 1)
52     ny_y = (center_y - seed_y) / np.size(Cropped, 0)
53     ny_w = width / np.size(Cropped, 1)
54     ny_h = height / np.size(Cropped, 0)
55     if (center_x + width/2) > (np.size(Cropped, 1) + seed_x) and (center_x - seed_x) < (np.size(Cropped, 1)):
56         # nytt uttrykk for bredde
57         ny_w = ((seed_x - np.size(Cropped, 1)) - (center_x - width/2))/np.size(Cropped, 1)
58         # flytter bbox center
59         x = (np.size(Cropped, 1) - seed_x) - (ny_w*np.size(Cropped, 1))/2
60         ny_x = x / np.size(Cropped, 1)
61     if (center_y + height/2) > (np.size(Cropped, 0) + seed_y) and (center_y - seed_y) < (np.size(Cropped, 0)):
62         ny_h = ((seed_y - np.size(Cropped, 0)) - (center_y - height / 2)) / np.size(Cropped, 0)
63
64     if center_x < seed_x or center_y < seed_y:
65         continue
66     elif center_x > (np.size(Cropped, 1) + seed_x) or center_y > (np.size(Cropped, 0) + seed_y):
67         continue
68     else:
69         Ofile.write(obj_id + " " + str(ny_x) + " " + str(ny_y) + " " + str(ny_w) + " " + str(ny_h))
70         Ofile.write("\n")
71         delete = False
72
73 if delete:
74     # Hvis ingen objekter i croppet bilde, eller alle objekters center er utenfor crop, slett croppet bilde
75     os.remove(path)
76     os.remove(Path)
77     File.close()
78     Ofile.close()
79
80 last = last + 1
81
82
```

Vedlegg C – CMake og C++ Kildekode

C.1 – CMakeLists.txt

```
1  #[[ CMakeLists.txt
2  Script for å generere "build-script" for prosjektet
3  Erlend Omland & Marthe Engvik - Med god hjelp fra Martin Vonheim Larsen
4  I forbindelse med Bacheloroppgave i Militære studer med ferdigstillelse i Elektronikk og Data ]]
5
6  cmake_minimum_required(VERSION 3.7)
7  project(Utkikk9000)
8
9  find_package(CUDA REQUIRED)
10 find_package(OpenCV REQUIRED COMPONENTS core highgui imgcodecs videoio)
11
12 add_library(darknet SHARED IMPORTED)
13
14 set(darknet_root /home/bachelor/darknet-master)
15
16 set_target_properties(darknet PROPERTIES
17     INTERFACE_INCLUDE_DIRECTORIES "${darknet_root}/src;${CUDA_INCLUDE_DIRS}"
18     IMPORTED_LOCATION "${darknet_root}/darknet.so"
19 )
20
21 add_executable(Utkikk9000
22     main.cpp
23     funksjoner.h)
24
25 set_target_properties(Utkikk9000 PROPERTIES
26     CXX_STANDARD_REQUIRED ON
27     CXX_STANDARD 14
28 )
29
30 target_compile_definitions(Utkikk9000
31     PRIVATE GPU
32 )
33
34 target_link_libraries(Utkikk9000
35     PRIVATE darknet
36     PRIVATE ${CUDA_LIBRARIES}
37     PRIVATE ${OpenCV_LIBRARIES}
38 )
```

C.2 – main.cpp

```
1  /*
2   * main.cpp
3   * hovedprogram i C++ applikasjon for å bruke darknet til å kjøre deteksjonsmodellen
4   * Erlend Omland & Marthe Engvik
5   * I forbindelse med Bacheloroppgave i Militære studier med fordypning i Elektronikk og Data
6   */
7
8 #include "funksjoner.h"
9
10 int main() {
11     std::string cfg = "/home/bachelor/darknet-master/cfg/yolov3-Utkikk9000.cfg";
12     std::string weights = "/home/bachelor/darknet-master/yolov3-Utkikk9000_v4.0_best.weights";
13     std::string names = "/home/bachelor/darknet-master/data/obj.names";
14     int gpu = 0;                                     //GPU index
15     float det_thresh = 0.5;                          //Detection threshold
16
17     Detector detector(cfg, weights, gpu);
18     auto obj_names = objects_names_from_file(names);
19
20     //Funksjonalitet for å kunne analysere utvalgte videofiler gjennom nettverket
21     std::string periscope[8] = {"AA-C0002_nosound.MP4", "BB-C0003_nosound.MP4", "CC-C0004_nosound.MP4",
22                               "DD-C0005_nosound.MP4", "EE-C0006_nosound.MP4", "GG-C0010_nosound.MP4",
23                               "HH-C0011_nosound.MP4", "Demov2.mp4"};
24     std::string path = "/home/bachelor/darknet-master/Periskop/";
25     path.append(periscope[5]);
26
27
28     cv::VideoCapture cap(0);                         //Åpner webkamera
29     cap.set(CV_CAP_PROP_FRAME_WIDTH, 1280);
30     cap.set(CV_CAP_PROP_FRAME_HEIGHT, 720);
31
32     while (cap.isOpened()){                         //Deteksjonsløkke
33         cv::Mat frame;
34         cap >> frame;
35         std::vector<bbox_t> result_vec;
36
37         auto start = std::chrono::steady_clock::now(); //Deteksjon og FPS utregning
38         result_vec = detector.detect(frame, det_thresh);
39         auto end = std::chrono::steady_clock::now();
40         std::chrono::duration<double> spent = end - start;
41         double detect_time = spent.count();
42         double fps = 1/detect_time;
43         std::ostringstream FPS;
44         FPS << "FPS: ";
45         FPS << std::setprecision(3);
46         FPS << fps;
47
48         draw_boxes(frame, result_vec, obj_names);      //Deteksjonsvisualisering i bilde
49         cv::putText(frame, FPS.str(), cv::Point(10,10), CV_FONT_HERSHEY_PLAIN, 1.0, cv::Scalar(0,255,0));
50         show_console_result(result_vec, obj_names);
51
52         cv::namedWindow("Utkikk9000", CV_WINDOW_NORMAL);
53         cv::setWindowProperty("Utkikk9000", CV_WND_PROP_FULLSCREEN, CV_WINDOW_FULLSCREEN);
54         cv::imshow("Utkikk9000", frame);
55
56         if(cv::waitKey(10) == 27){                     //Trykk "esc" for å avslutte programmet
57             break;
58         }
59
60     }
61
62     return 0;
63 }
```

C.3 – funksjoner.h

```
1  /* funksjoner.h
2   Header som inneholder funksjoner for bruk av darknet og trenet modell i C++ applikasjon.
3   Videre inneholder filen funksjoner for å visualisere deteksjoner.
4   Erlend Omland & Marthe Engvik - Med god hjelp fra Martin Vønheim Larsen
5   I forbindelse med Bacheloroppgave i Militære studer med fordypning i Elektronikk og Data
6   */
7
8 #ifndef UTKIKK9000_FUNKSJONER_H
9 #define UTKIKK9000_FUNKSJONER_H
10
11 #include <iostream>
12 #include <vector>
13 #include <string>
14 #include <thread>
15 #include <iomanip>
16 #include <sstream>
17 #define OPENCV
18 #include <yolo_v2_class.hpp>
19
20 // Visualiserer klasse som enten military/civillian, relativ kurs skal representeres med pil
21 std::string find_name(int id){
22     std::string name;
23     switch(id){
24         case 0: {
25             name = "Civillian";
26             break;
27         }
28         case 1: {
29             name = "Civillian";
30             break;
31         }
32         case 2: {
33             name = "Civillian";
34             break;
35         }
36         case 3: {
37             name = "Military";
38             break;
39         }
40         case 4: {
41             name = "Military";
42             break;
43         }
44         case 5: {
45             name = "Military";
46             break;
47         }
48     }
49     return name;
50 }
51
52 //Tegner bokser rundt detekterte objekter. Videre beregnes relativ peiling, samt relativ kurs.
53 void draw_boxes(cv::Mat mat_img, std::vector<bbox_t> result_vec, std::vector<std::string> obj_names) {
54     int const colors[6][3] = {{1, 0, 1},
55                             {0, 0, 1},
56                             {0, 1, 1},
57                             {0, 1, 0},
58                             {1, 1, 0},
59                             {1, 0, 0}};
60 }
```

```

61     for (auto &i : result_vec) {
62         cv::Scalar color = obj_id_to_color(i.obj_id);
63         cv::rectangle(mat_img, cv::Rect(i.x, i.y, i.w, i.h), color, 2);
64         if (obj_names.size() > i.obj_id) {
65             std::ostringstream presisjon;
66             presisjon << std::setprecision(2);
67             presisjon << i.prob;
68             std::string obj_name = find_name(i.obj_id) + " " + presisjon.str();
69             if (i.track_id > 0) obj_name += " - " + std::to_string(i.track_id);
70             cv::Size const text_size = getTextSize(obj_name, cv::FONT_HERSHEY_COMPLEX_SMALL, 1.2, 2, 0);
71             int const max_width = (text_size.width > i.w + 2) ? text_size.width : (i.w + 2);
72             cv::rectangle(mat_img, cv::Point2f(std::max((int)i.x - 1, 0), std::max((int)i.y - 30, 0)),
73                         cv::Point2f(std::min((int)i.x + max_width, mat_img.cols - 1),
74                                     std::min((int)i.y, mat_img.rows - 1)),
75                         color, CV_FILLED, 8, 0);
76             putText(mat_img, obj_name, cv::Point2f(i.x, i.y - 10), cv::FONT_HERSHEY_COMPLEX_SMALL, 0.7,
77                     cv::Scalar(0, 0, 0), 2);
78             if (i.obj_id == 0 || i.obj_id == 3){ //Relative kurs beregnes her, videre visualiseres dette i bildet.
79                 cv::arrowedLine(mat_img, cv::Point2f(i.x + 50, i.y - 40), cv::Point2f(i.x, i.y - 40),
80                               cv::Scalar(0, 0, 0), 3, 8, 0, 0.2);
81             }
82             else if (i.obj_id == 1 || i.obj_id == 4){
83                 cv::arrowedLine(mat_img, cv::Point2f(i.x, i.y - 40), cv::Point2f(i.x + 50, i.y - 40),
84                               cv::Scalar(0, 0, 0), 3, 8, 0, 0.2);
85             }
86             int bearing; //Relativ peiling i forhold til bildesenter beregnes, videre visualiseres dette i bildet.
87             if ((i.x + i.w/2) < (mat_img.cols/2)){
88                 cv::drawMarker(mat_img, cv::Point2f(i.x + i.w/2, mat_img.rows - 15), cv::Scalar(0, 255, 0),
89                               cv::MARKER_TRIANGLE_DOWN, 16, 2, 8);
90                 int bearing = int(90 - cv::fastAtan2(mat_img.rows - (i.y + i.h/2), mat_img.cols/2 - (i.x + i.w/2)));
91                 std::string Bearing = std::to_string(bearing);
92                 cv::putText(mat_img, Bearing, cv::Point2f(i.x + i.w/2 - (4 + 2*Bearing.length()), mat_img.rows - 27),
93                               cv::FONT_HERSHEY_COMPLEX_SMALL, 0.7, cv::Scalar(0, 255, 0), 1);
94             }
95             else if ((i.x + i.w/2) > (mat_img.cols/2)){
96                 cv::drawMarker(mat_img, cv::Point2f(i.x + i.w/2, mat_img.rows - 15), cv::Scalar(0, 0, 255),
97                               cv::MARKER_TRIANGLE_UP, 16, 2, 8);
98                 int bearing = int(90 - cv::fastAtan2(mat_img.rows - (i.y + i.h/2),(i.x + i.w/2) - mat_img.cols/2));
99                 std::string Bearing = std::to_string(bearing);
100                cv::putText(mat_img, Bearing, cv::Point2f(i.x + i.w/2 - (4 + 2*Bearing.length()), mat_img.rows - 27),
101                               cv::FONT_HERSHEY_COMPLEX_SMALL, 0.7, cv::Scalar(0, 0, 255), 1);
102            }
103        }
104    }
105 }
106 }
107 }

108 ////////////////////////////////////////////////////////////////////FUNKSJONER HENTET FRA ÅPEN KILDEKODE/////////////////////////////////////////////////////////////////
109 void show_console_result(std::vector<bbox_t> const result_vec, std::vector<std::string> const obj_names) {
110     for (auto &i : result_vec) {
111         if (obj_names.size() > i.obj_id) std::cout << obj_names[i.obj_id] << " - ";
112         std::cout << "obj_id = " << i.obj_id << ", x = " << i.x << ", y = " << i.y
113         << ", w = " << i.w << ", h = " << i.h
114         << std::setprecision(3) << ", prob = " << i.prob << std::endl;
115     }
116 }
117 }

118 std::vector<std::string> objects_names_from_file(std::string const filename) {
119     std::ifstream file(filename);
120     std::vector<std::string> file_lines;
121     if (!file.is_open()) return file_lines;
122     for(std::string line; getline(file, line);) file_lines.push_back(line);
123     std::cout << "object names loaded \n";
124     return file_lines;
125 }
126 }

127
128
129
130 #endif //UTKIKK9000_FUNKSJONER_H
131

```

Vedlegg D – Konfigurasjonsfiler

Diverse datafiler som kreves for å kjøre vår modell i darknet

D.1 - obj.data

D.2 - obj.names

D.3 - test.txt

D.4 - train.txt

D.5 - yolov3-Utkikk9000.cfg

Vedlegg E – Datasett

Datasettet produsert til oppgaven i sin helhet. 7643 .jpg og 7643 .txt filer

Vedlegg F – Vektfiler

Beste vekter fra hver fase av utvikling

F.1 - v1.0

F.2 - v2.0

F.3 - v3.0

F.4 - v4.0

Vedlegg G – Videofiler fra testing

Video fra tester gjennomført med systemet

G.1 - Forhold_dårlige.avi

G.2 - Forhold_optimale.avi

Vedlegg H – Åpen-kildekode

Dette er de ulike programvarene som er benyttet i utviklingen av klassifiseringssystemet. Programvare listet under er åpen-kildekode.

Darknet - <https://github.com/AlexeyAB/darknet>

Yolo_mark - https://github.com/AlexeyAB/Yolo_mark

OpenCV 3.4.0 - <https://opencv.org/releases.html>

CUDA 9.1 - <https://developer.nvidia.com/cuda-91-download-archive>

ScrapeImages.py - <https://gist.github.com/genekogan/ebd77196e4bf0705db51f86431099e57>

