Bacheloroppgave

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

Simen Andre Nørstebø
Espen Myrum

Automatisk gjenkjenning av settefisk

Bacheloroppgave i Bachelor ingeniør - Data
Veileder: Marius Pedersen, Sony George
Mai 2019

NTNU
Kunnskap for en bedre verden

Simen Andre Nørstebø
Espen Myrum

Automatisk gjenkjenning av settefisk

Bacheloroppgave i Bachelor ingeniør - Data
Veileder: Marius Pedersen, Sony George
Mai 2019

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

NTNU
Kunnskap for en bedre verden

Sammendrag av Bacheloroppgaven

Tittel:	**Automatisk gjenkjenning av settefisk**
Dato:	20.05.2019
Deltakere:	Espen Myrum Simen Andre Nørstebø
Veiledere:	Sony George Marius Pedersen
Oppdragsgiver:	Norsk Institutt for Naturforskning
Kontaktperson:	Jon Museth
Nøkkelord:	Maskinlæring, Datasyn, Dyp læring, IMT, Klassifisering, Fisk
Antall sider:	68
Antall vedlegg:	3
Tilgjengelighet:	Åpen

Sammendrag:

Norsk institutt for naturforskning ønsket å se om det var mulig å automatisk gjenkjenne om en undervannsvideo viser en settefisk eller villfisk. I denne rapporten skal vi se nærmere på ulike metoder for å løse dette problemet med maskinlæring eller datasyn. Vi har sett hvor viktig det er med et bra datasett, og bevist at teknikkene vi har brukt har hatt en positiv innvirkning på resultatene. Vi trente et Konvolusjonelt nevralt nettverk (KNN)- og Error correcting ouput codes (ECOC)-modell til ca. 99% nøyaktighet. Til slutt implementerte vi modellene inn i en fungerende applikasjon som klassifiserer video av kultivert ørret og villfisk gjennom bildeuthenting og -klassifisering.

Summary of Graduate Project

Title:	**Automatic detection of hatched fish**
Date:	20.05.2019
Authors:	Espen Myrum Simen Andre Nørstebø
Supervisor:	Sony George Marius Pedersen
Employer:	Norsk Institutt for Naturforskning
Contact Person:	Jon Museth
Keywords:	Machine Learning, Computer Vision, Deep Learning, IMT, Classification, Fish
Pages:	68
Attachments:	3
Availability:	Open

Abstract: Norwegian Institute for Nature Research wanted to see if it was possible to automatically detect if a fish on an underwater video is from a hatchery or is wild. In this thesis we looked into different methods for solving this problem using machine learning and computer vision. We have seen the importance of a good dataset, and shown that the techniques we used have had a positive impact on the results. We trained a Convolutional Neural Network (CNN)- and Error correcting ouput codes (ECOC)-model to an accuracy of approx. 99%. Finally we implemented these models into a working application which classifies videos of hatched and wild trout through image extraction and classification.

Forord

Vi ønsker å rette en stor takk til Norsk institutt for naturforskning (NINA) for at vi fikk prøve oss på denne oppgaven. Det har vært veldig spennende og lærerikt, og vi kunne ikke bedt om en bedre oppgave. Vi håper resultatene fra denne oppgava vil være nyttig for NINA, og at applikasjonen kan bidra til å automatisere klassifiseringen.

Vi vil også takke veilederene våre prof. Marius Pedersen og prof. Sony George for jevnlig assistanse og veiledning, hvor vi har fått konstruktiv kritikk og tilbakemeldinger som har hjulpet oss utrolig mye i løpet av hele prosjektet. En stor takk også til Overingeniør Binu Melit Devassy ved NTNU Colorlab for tilgang til en ekstra datamaskin i forbindelse med maskinlæring.

Innhold

Figurer

Tabeller

Listings

Ordliste

augmentering Augmentering betyr å forstørre eller utvide. I denne sammenhengen vil det si å utvide et datasett, altså å bruke et bilde til å lage et nytt bilde med ny informasjon. For eksempel ved å speilvende et bilde.. , 23

Dropout Dropout er en regulariserende teknikk for neurale nettverksmodeller. , 17

Gauss Carl Friedrich Gauss, tysk matematiker, astronom, geodet og fysiker. Regnes som en av de største matematikerne i historien.. , 26

Matlab Matlab eller Matrix Laboratory er både et omfattende matematikkprogram og et eget programmeringsspråk for scripting utviklet av MathWorks. Det er modulært og kan utvides med såkalte verktøykasser for å legge til funksjonalitet. De mange bruksområdene har gjort det til en yndling for ingeniører.. , 2, 20, 32, 41, 43, 55

OpenCV OpenCV er et "open-source" BSD-lisensiert bibliotek som inneholder flere hundre datasynalgoritmer. Det finnes til bl.a. Python og C++. , 21, 55

overfitting Overfitting eller overtrening er når modellen man trener blir for godt trent på treningsdataene. Dette fører til dårligere resultat på ny data.. , 16, 17

Overføringslæring Overføringslæring (eng: transfer learning) vil si å ta et allerede trent nettverk og tilpasse til et nytt klassifiseringsproblem eller ny data.. , 16, 30

Python Python er et scripting-/programmeringsspråk som er designet med fokus på lesbarhet og simplisitet. Kode skrevet i Python blir oversatt, som vil si at koden oversettes til maskinkode under kjøring, i motsetning til f.eks. C++ hvor koden må kompileres før kjøring av programmet.. , 2, 20, 21, 37

ReLu ReLu er en type aktiveringsfunksjon, som er den vanligste å benytte for KNN. Den er ikke lineær og har samme fordelene som Sigmoid, men med bedre ytelse. , 17

Scrumban Scrumban er en hybrid av Scum og Kanban.. , 6

Sigmoid Sigmoid er en aktiveringsfunksjon som tar inn en reell verdi og som har verdiene 0 og 1 som utdata. Den er ikke lineær, men er differensierbar, monotonisk og har et fast utgangsområde. , 17

Taiga.io Scrum-verktøy som hjelper gruppen med å planlegge, estimere og gjennomføre prosjektet. , 6

Tanh Tanh er som en logistisk Sigmoid, men bedre. Funksjonen operer i intervallet (-1 til 1). , 17

Toggl System som benyttes av gruppen for å føre timelister med beskrivelser om hva som har blitt gjort. , 58

Akronymer

AdaDelta Adadelta. , 19

AdaGrad Adaptive Gradient Algorithm. , 19

Adam Adaptive moment estimation. , 18, 33

CLAHE Contrast-limited adaptive histogram equalization. , 23, 29, 33, 43, 56

CNN Convolutional Neural Network. ii,

ECOC Error correcting ouput codes. i, ii, , 32, 36, 43, 44, 64

IDI Institutt for Datateknologi og Informatikk. , 3, 5

ILSVRC ImageNet Large Scale Visual Recognition Challenge. , 17, 18

JPEG/JPG Joint Photographic Experts Group. , 12, 56, 65

KNN Konvolusjonelt nevralt nettverk. i, , xi, 15, 17, 18, 32, 36, 43, 44, 64, 65

MPEG Moving Picture Experts Group. , 12

NINA Norsk institutt for naturforskning. iii, , 1, 3, 44, 55, 64, 65

NTNU Norges Teknisk-Naturvitenskapelige Universitet. , 2, 3, 5, 20, 55, 60

PNG Portable Network Graphics. , 12, 56

RGB Rød, grønn og blå. , 12, 24, 25, 57

RMSprop Root Mean Square Propagation. , 19, 33

SGD Stochastic Gradient Descent. , 18, 19

SGDM Stochastic Gradient Descent with Momentum. , 19, 33, 35, 43

SVM Support vector machine. , 32

1 Innledning

1.1 Bakgrunn

Bachelor i ingeniørfag data på Gjøvik avsluttes med bacheloroppgaven. Denne oppgaven skal være forankret i reelle problemstillinger fra samfunns- og næringsliv eller forsknings- og utviklingsarbeid og bidra til innføring i vitenskapsteori og metode, samt integrere viktige deler av programmets faglige innhold. Prosjektet skal bunne ut i en rapport som dokumenterer dette arbeidet på en strukturert måte. I den forbindelse skal vi utføre et prosjekt for Norsk institutt for naturforskning (NINA). NINA er en uavhengig stiftelse som forsker på natur og samspillet natur - samfunn.

Hunderfossen kraftverk stod ferdig i 1964. Det ble etablert ei fisketrapp for å få gytefisk av ørret forbi den store kraftverksdemningen og det ble også etablert ei fysisk felle midt i fisketrappa slik at man kunne overvåke fiskeoppgangen. Dette har gitt viktige data om utviklingen i ørretbestanden, men samtidig har undersøkelser vist at fangst av ørret i fella og håndtering av fisken (bl.a. måling og merking) kan ha negative konsekvenser for fisken. Det har vært et ønske om å raffinere overvåkingsmetoden, og i 2016 ble det installert en automatisk fisketeller slik at fisken kan svømme fritt gjennom fisketrappa. Dette er også et framskritt for dyrevelferden og krever hverken fangst eller håndtering av fisken som passerer fisketrappa.

(a) Villfisk

(b) Settefisk

Figur 1: Villfisk og settefisk

Her har vi et godt eksempel på en bilderamme med villfisk og settefisk. På figur 1a ser vi "fettfinna" som er inni den røde ringen, mens på figur 1b så er den klipt av.
Hos all kultivert ørret fra Hunderfossen er "fettfinna" klipt av før utsetting som 2-åringer. Denne vokser aldri ut igjen, og brukes til å skille dem fra villfisken. Etter noen år i Mjøsa kommer disse tilbake som gytefisk til Gudbrandsdalslågen. Det er enkelt å skille naturlig rekruttert ørret (villfisk) fra kultivert ørret (settefisk) hvis man går gjennom alle videosnuttene manuelt. Hver fisk blir registrert med et "scannerbilde" og en

videosnutt på ca. 24 sekunder. Dette er viktige data fordi man ønsker å øke den naturlige produksjonen av villfisk i elva.

1.1.1 Formålet

Målet for bacheloroppgaven er å implementere datasynteknikker og eventuelt maskinlæring for å skille naturlig rekruttert ørret og kultivert ørret ut i fra video fra fisketrappa i Hunderfossen kraftverk i Gudbrandsdalslågen.

1.1.2 Forutsetning for oppgaven

Vi studerer begge ingeniørfag - data og har relativt lik fagbakgrunn innenfor dette. Det er i stor grad de obligatoriske fellesemnene vi har fått bruk for i denne oppgaven. Programmering på alle nivå, statistikk og økonomi, ingeniørrollen, ingeniørfaglig systememne og systemutvikling er alle emner som har hjulpet oss i gjennomføringen.
Det vi derimot ikke har hatt, som ville vært til stor hjelp er bla. emnene Computer Vision og Kunstig Intelligens. Disse emnene fokuserer i stor grad på hovedtemaene i denne oppgaven, og de hadde dermed vært god forkunnskap å inneha. Som en effekt av dette ble vi nødt til å sette av en god del tid til å lese oss opp på disse temaene i forkant av utviklingen. Det ble også brukt mye tid på dette underveis i oppgaven, etterhvert som vi kom dypere inn i fagområdene.
Tidlig i prosjektet skjønte vi også at ny teknologi måtte læres i denne forbindelse. Programmeringsspråket Python, og Matlab virket å være de to mest utbredte for løsing av liknende oppgaver. Begge er heldigvis forholdsvis enkle i forhold til C++ som vi lærte i programmeringsemnene på NTNU, og med god dokumentasjon.

1.2 Omfang

1.2.1 Fagområde

Oppgaven gir oss frihet til å benytte datasyn og maskinlæring for å lage en programvare som filtrerer om det er kultivert ørret eller villfisk. I den forbindelse kreves det kunnskap i en del ukjente programmeringsspråk. I oppstartsfasen vil vi se nærmere på hvilke teknikker som gir best utbytte for oss i vårt prosjekt. Vi vil også se på de mest populære programmeringsspråkene brukt innen datasyn, MATLAB og Python, før vi avgjør hva vi skal bruke. Vi skal skille ut bilder fra datasettet vi blir tildelt og bygge videre på dette for prosjektet vårt.

1.2.2 Oppgavebeskrivelse

I mange elver og vann i Norge brukes settefisk for å opprette og opprettholde fiskebestander. Settefisk er kunstig klekkede fiskeunger som fôres en tid før de settes ut i ferskvann eller sjøen for videre vekst. Før disse settes ut så fjernes fettfinnen på fisken, dette gjøres for å kunne gjenkjenne settefisk fra villfisk. Norsk Institutt for Naturforskning (NINA) har montert kamera i flere norske elver for å kunne overvåke fiskebestander. I dag er gjenkjenningen av settefisk en manuell oppgave.

I denne oppgaven skal vi utvikle en automatisk metode for gjenkjenning av settefisk. Gjenkjenningen baseres på undervannsbilder (video). Disse bildene er tatt under forskjellige forhold (forskjellige lysforhold, farge på vannet, luftbobler, etc.). Gjenkjenningen må være robust og fungere under alle slags forhold.

1.2.3 Avgrensning

Vi ble nødt til å avgrense prosjektet noe da vi opprinnelig skulle være tre medlemmer, men etter en uforutsett hindring endte vi opp med å bli kun to.
Oppdragsgiver ønsket å se om det var mulig gjenkjenne den samme ørreten som var i fisketrappen hvis den kommer tilbake noen år senere. Vi ble enige sammen med oppdragsgiver at det ble for krevende og at vi i første omgang kun skulle fokusere på gjenkjenning av kultivert ørret og villfisk.

1.3 Mål og rammer

1.3.1 Målgruppe

Målgruppe rapport

Rapportens målgruppe er hovedsaklig studenter og ansatte under Institutt for Datateknologi og Informatikk (IDI) ved NTNU, men også andre interesserte innen for eksempel maskinlæring, datasyn og bildebehandling. Den vil også gi god innsikt for de som vil videreutvikle produktet, eller studenter med liknende prosjekter.

Målgruppe produkt

Produktets målgruppe er i første omgang de ansatte hos NINA som vil bruke dette til å klassifisere fisk som går gjennom fisketelleren i Hunderfossen. Men da slik måling gjøres i flere elver der det demmes opp kan det bli aktuelt å implementere dette på disse stedene. Overføringsverdien til andre elver og prosjekter er derfor potensielt stor.

1.3.2 Hva må læres?

I dette prosjektet er nesten alt nytt for oss. Vi har noe erfaring fra tidligere i forbindelse med utviklingsmodeller, men det som kreves for å løse denne oppgaven har vi ikke vært borti tidligere. Læringsstoffet blir datasyn og maskinlæringsteknikker.

1.3.3 Krav

Oppdragsgiver har ingen spesifikke krav til oppgaven, vi har fastsatt følgende krav for oppgaven:

• Videoklippene skal kun være ørret og sortert etter settefisk/villfisk av oppdragsgiver.

• Videoklippene skal kun vise en ørret om gangen som går gjennom fisketrappen i Hunderfossen.

• Utviklingsmodellen skal være tilpasningsdyktig slik at vi kan legge til funksjonalitet dersom vi får tid til dette.

1.4 Øvrige roller

Vi har følgende roller for vårt prosjekt:

Interne roller:	
Gruppeleder	Espen Myrum
Kommunikasjonsansvarlig	Simen Nørstebø
Eksterne roller:	
Oppdragsgivers kontaktperson	Jon Museth
Veileder	Sony George
Veileder	Marius Pedersen

1.5 Rapportens struktur

I denne rapportens skal språket være tilpasset medstudenter innenfor Institutt for Datateknologi og Informatikk (IDI) ved NTNU, men også andre interesserte som har kunnskap om velkjente "ord og uttrykk".
Skriverktøyet vi benytter for rapporten er gjort i LaTeX som er et verktøy for skriving av alle slags dokumenter.

Rapporten er delt opp i ni hovedkapittel:

- **Kapittel 1: Innledning**
  Inneholder introduksjon til oppgaven og rapport, samt hva som er motivasjonen for å gjøre denne oppgaven, litt om bakgrunn og organisering av prosjektet.

- **Kapittel 2: Arbeidsmetode**
  Beskriver hvordan gruppen planlegger å jobbe med bacheloroppgaven og hvilke rammer prosjektet har.

- **Kapittel 3: Bakgrunn**
  Inneholder teknisk informasjon rundt datasyn og maskinlæring. Forteller om forskjellige læringsalgoritmer i maskinlæring vi har hatt nytte av.

- **Kapittel 4: Datasett og preprossesering**
  Beskriver hvordan vi har arbeidet med datasettet vi har fått utdelt, hva vi har gjort og hvordan vi har brukt det.

- **Kapittel 5: Implementering og realisering**
  Inneholder hoveddelen av prosjektet, tar for seg store deler av utviklingsprosessen hvor vi har trent nettverk, samt en liten innføring i hvordan applikasjonen fungerer.

- **Kapittel 6: Resultater**
  Inneholder resultater vi har fått i løpet av prosjekttiden.

- **Kapittel 7: Utførelse**
  Inneholder sammendrag fra hver enkel periode, som oftes sprinter i prosjektet.

- **Kapittel 8: Drøfting**
  Inneholder drøfting og evaluering av gruppearbeidet og arbeidsprosessen vår.

- **Kapittel 9: Konklusjon og videre arbeid**
  Inneholder en konklusjon av prosjektet vårt samt tips til videre arbeid.

2 Arbeidsmetode

Dette kapittelet beskriver hvordan vi planla å jobbe med bacheloroppgaven vår.

2.1 Valg av utviklingsmodell

Under planlegging av prosjektet valgte vi utviklingsmetode for vårt prosjekt.
Valget vårt falt på Scrumban for utvikling av prosjektet. Siden det er en hybrid av Scrum og Kanban gir det prosjektet vårt fleksibilitet til å tilpasse seg og gjøre endringer underveis om vi får tid til å legge til mer funksjonalitet, eller om det oppstår uforutsette utfordringer.
Denne modellen gir oss struktur og prosesser fra Scrum og den åpenheten som Kanban tilbyr. Kanban tilbyr oss Kanban Board som gir oss lett oversikt over vår egen utvikling, om hva som har blitt gjort og hva som skal gjøres. Vi skulle benytte oss av Taiga.io som er et scrum-verktøy for å planlegge, estimere og gjennomføre prosjektet [1].

Parprogrammering lar oss begge følge med på hva som blir gjort, når det blir gjort, samt at vi kan gi hverandre innspill under utviklingen. Dermed trenger vi ikke å vente helt til gjennomgangen av det som har blitt gjort med å gjøre mindre endringer, eller foreslå forbedringer.

Argumenter som var med på avgjørelsen vår:

- Vi er en liten gruppe på to medlemmer.
- Vanskelig å tidsestimere utfordringer i prosjektet, smidig utvikling vil være en en fordel.
- Prosjektet har ingen sluttilstand, det er mulig å tilføye og utvikle mer.

Gruppen drøftet disse utviklingsmodellene:

Scrum

En smidig utviklingsmodell som tilbyr en gitt struktur med sprinter. Sprinter deles igjen inn i mindre oppgaver, og har et fast intervall som bestemmes ved oppstart. Selve metoden krever mye arbeid med organisering og dokumentering, da vi er to medlemmer vil dette ta mye tid og vi trenger den arbeidskraften vi har til rådighet.

Kanban

Kanban er en enkel modell som benytter Kanban Board hvor arbeidet blir delt opp og organisert i forskjellige tilstander. På denne tavlen vil vi få en god oversikt over hva som må gjøres og hva som har blitt gjort.

Parprogrammering

Parprogrammering er en smidig modell som baserer seg på to programmerere som deler en enkelt datamaskin. Programmereren på tastaturet kalles vanligvis "føreren", den andre er aktivt involvert i programmeringen, men fokuserer mer på den totale utførelsen som kalles "navigatøren".

2.2 Dokumentasjonskrav

Arbeidsprosess

Alt av arbeid og endringer vi gjør underveis skal dokumenteres slik at det blir enklere å se tilbake på det vi har arbeidet med gjennom hver enkelt sprint. Vi benyttet oss av "Toggl"[2] til å føre timelister med beskrivelser av hva som har blitt gjort, og møtereferater og notater fra sprinter har blitt lagret i "Google Drive"[3].

Kildekode

For kildekode benytter vi Bitbucket[4] da dette tillater oss å jobbe fra eksterne lokasjoner i perioder hvor vi ikke benytter parprogrammering, samtidig som det gir en ekstra forsikring mot tap av data som er lastet opp. Det sikrer også at begge til enhver tid har siste versjon av koden.

I koden vil vi benytte beste praksis med tanke på kommentering, slik at leseligheten er god samt at det som utføres er lettoppfattelig. Det vil si at vi kommenterer idet vi skriver koden, da dette er både utfordrende og tidkrevende i ettertid.

Milepæler

I planleggingsfasen ble det satt opp fem milepæler. Etter den første milepælen har vi jobbet i flere sprinter for å kunne nå de disse målene. Vi var usikre i planleggingsfasen hva som måtte til for å nå disse da dette var et nytt stoff for oss, derfor ble det satt opp god tid for å kunne nå de.

Forprosjekt (01.02.2019)
Prosjektplan, grupperegler og prosjektavtale være levert inn.

Egenstudie fullført (18.02.2019)
Tilegnet oss nødvendig kunnskap for fortsettelse av prosjekt.

Fungerende prototype (01.04.2019)
Skal ha en funksjonibel prototype som klarer å skille forskjellen på en kultivert ørret og villfisk på grunnlag av bildene vi har tatt fra videoklippene.

Automatisk gjenkjenning (29.04.2019)
Vi skal ha utvilket en programvare som gjenkjenner om fisken er en kultivert ørret eller en villfisk. Resten av prosjekttiden vil bli brukt til ferdigstilling av rapporten.

Innlevering av prosjekt (20.05.2019)
Frist for innlevering av rapporten.

Veiledningsmøter

I hvert møte med veileder dokumenterte vi det vi gjennomgikk og videre arbeid som skulle gjennomføres ble noterte ned. Vi noterte også ned spørsmål før kommende veiledningsmøte slik at vi kom forberedt og vi fikk utnyttet veiledningsmøtet. Dokumentene vi skriver lagres på "Google Drive" og referat fra hvert møte finner du i vedlegg i A

2.3 Risikoanalyse

Risikomatrise					
Sannsynlighet	Svært sannsynlig	Moderat	Høy	Kritisk	Kritisk
	Sannsynlig	Lav	Moderat	Høy	Kritisk
	Lite sannsynlig	Lav	Lav	Moderat	Høy
	Usannsynlig	Lav	Lav	Lav	Moderat
		Ubetydelig	Betydelig	Alvorlig	Svært alvorlig
	Konsekvens				

Figur 2: Risikomatrise

Beskrivelse av risikomatrisen

Vi har i vår risikomatrise valgt å bruke fire nivåer av sannsynlighet, fra usannsynlig til svært sannsynlig. Det samme har vi gjort for konsekvens, som rangeres fra ubetydelig til svært alvorlig. Tilsvarende ender dette opp i fire nivåer av risiko, fra lav til kritisk på øverste nivå. Ved å multiplisere sannsynlighet med konsekvens ser man enkelt ut fra matrisen på figur 2 hvor risikonivået ligger. Dette danner grunnlaget for risikoanalysen vår.

Hendelse	Sannsynlighet	Konsekvens	Risiko
Sykdom blant prosjektmedlemmer	Sannsynlig	Alvorlig	Høy
Prosjektmedlem forlater prosjektet grunnet sykdom skade, sykdom eller andre årsaker	Usannsynlig	Svært Alvorlig	Moderat
Får ikke tilgang til videomaterialet	Usannsynlig	Svært Alvorlig	Moderat
Klarer ikke fullføre prosjektet innen gitt tidsramme	Sannsynlig	Svært Alvorlig	Kritisk
Videomateriale vi får er av for dårlig kvalitet	Sannsynlig	Alvorlig	Høy
Uenighet i gruppa ved avgjørelser	Sannsynlig	Ubetydelig	Lav
Tap av kildekode av forskjellige grunner	Lite sannsynlig	Svært Alvorlig	Høy
Får ikke tilgang til nok videomateriale	Lite sannsynlig	Alvorlig	Moderat

Figur 3: Risikovurdering

Beskrivelse av risikovurderingen

Vi har kommet opp med åtte hendelser vi ser som svært relevante for vårt prosjekt. For hver av disse har vi identifisert sannsynligheten og konsekvensen om disse skulle inntreffe. Videre har vi ved bruk av risikomatrisen kommet fram til risikoen for den enkelte hendelsen.

Hendelse	Tiltak
Sykdom blant prosjektmedlemmer	Påse at medlemmene får nok søvn, mat og drikke under prosjektets varighet. Ved sykdom under parprogrammering fortsetter arbeidet alene, og felles gjennomgang av koden vil skje så fort medlemmet er tilbake.
Prosjektmedlem forlater prosjektet grunnet sykdom skade, sykdom eller andre årsaker	Da vi allerede har mistet ett medlem og endte opp som en gruppe på to, er det kritisk om én til måtte forlate prosjektet. I samtale med veileder, samt programansvarlig må vi se på om det er mulig å ytterligere redusere omfanget av oppgaven.
Får ikke tilgang til videomaterialet	Uten videomaterialet har vi ingen oppgave, så tilgang til dette er kritisk for prosjektets gang. Instituttleder Marius Pedersen har fått noe data i forbindelse med utlysning av oppgaven, dette kan vi bruke hvis det tar for lang tid å få direkte fra NINA.
Klarer ikke fullføre prosjektet innen gitt tidsramme	Følge utviklingsmetoden og integrere denne på en strukturert måte, samt god planlegging og estimering av arbeidet. Funksjonaliteten vil implementeres i moduler, slik at vi først får grunnfunksjonaliteten opp og går før vi fokuserer på nye funksjoner.
Videomaterialet vi får er av for dårlig kvalitet	Bildebehandlingsverktøy må brukes for å forsterke kontraster, fjerne uønskede elementer, samt redusere støy.
Uenighet i gruppa ved avgjørelser	Da vi er bare to medlemmer må vi ved større uenigheter ta dette opp med veileder slik at vi får et tredje synspunkt, og velger den beste løsningen for alle parter. Hvis begge løsninger er likeverdige kan avgjørelsen tas ved tilfeldig trekning/myntkast.
Tap av kildekode av forskjellige grunner	Ha gode rutiner for lagring og oppbevaring av data. Alt skal lagres både lokalt og i Github/Bitbucket, slik at vi ved systemkrasj eller andre uforutsette hendelser mister minimalt med arbeid.
Får ikke tilgang til nok videomateriale	Det er flere metoder for å utvide et datasett bestående av bilder. For eksempel ved å rotere, skalere, vende eller tilføre et bilde støy kan man mangedoble datasettet. I verste fall må vi gå bort fra maskinlæring og bruke enklere datasynteknikker.

Figur 4: Risikotiltak

3 Bakgrunn

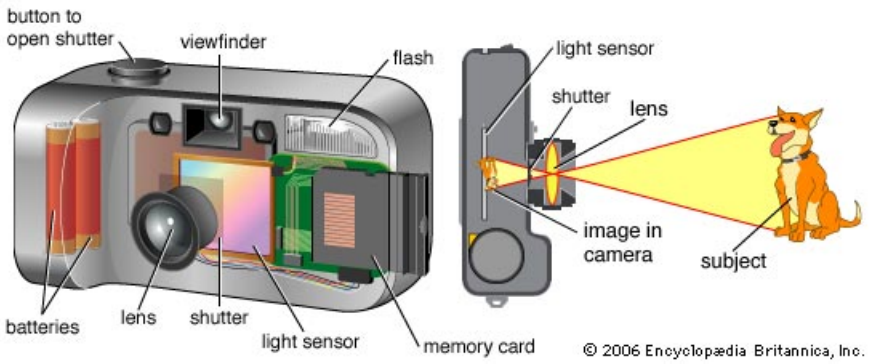
For å kunne lese og tolke resten av rapporten kreves det grunnleggende kunnskap om det faglige innholdet. Dette kapittelet er delt opp i datasyn og maskinlæring. Kapittelet beskriver kort teori som kreves for å forstå resten av rapporten.

3.1 Datasyn

Datasyn eller computer vision er en vitenskap med et mål om å gi tilsvarende eller bedre syn enn mennesker til en datamaskin. Dette innebærer automatisk innhenting, analyse og forståelse av nyttig data fra ett enkelt bilde, eller en sekvens av bilder [5].

3.1.1 Hvordan fungerer et kamera?

Prosessen for hvordan er kamera fungerer er som følger:
Når du tar bildet, slår lyset på den digitale sensordelen som "CCD" og "CMOS", som er laget av millioner av piksler, lagt ut i en rekke rader og kolonner (eksempel: 3000 x 1500 piksler oppløsning). Disse små lyspunktene som danner bildet, kommer gjennom fargefiltre over sensorene, og deretter konverterer sensorbrikken bildet fra lysbølger til analogt elektrisk signal, som deretter sendes til Analog-Digital-Converter. Andre elektroniske filtre justerer hvitbalansen, fargen og andre parametre. Deretter lagres det digitale bildet i bufferminnet til det blir fullt og deretter skrevet på lagringsmediet [6].



Figur 5: Kamera [6]

På figur 5, ser vi hvordan et kamera fungerer, selv om det ikke er et sånt type kamera som blir benyttet i oppgaven vår er selve kamerafunksjonen den samme.

3.1.2 Digitale bilder

Digitale bilder leses av en datamaskin som en strøm av numre. De består av piksler, og hver piksel kan ha flere lag med informasjon eller farger, kalt kanaler, som tilsammen utgjør et bilde som man ser på skjermen. Et bilde vil dermed leses som en matrise med antall piksler i x- og y-planet (bredde og høyde) og antall kanaler i hver piksel. Det er

denne matrisen som endres ved behandling av et bilde, og det er denne som analyseres av datasyn-applikasjoner. Et vanlig fargebilde (JPEG/JPG/PNG) har som regel tre kanaler; Rød, grønn og blå (RGB). Dermed har hver piksel tre sett med verdier. Dette er som regel 8 bits-verdier som gir verdier fra 0-255, og disse tre kanalene gir et fargespekter på rundt 16 millioner farger. I sort/hvitt benyttes kun en kanal, og dermed kun 8 bits, som gir 256 forskjellige toner.

3.1.3 Digital video - MPEG

Digital video håndteres på samme måte som digitale bilder.
Moving Picture Experts Group (MPEG) er et videoformat og en standard for videokomprimering [7]. MPEG har definert en rekke videoformater:

- **MPEG-1**
  Innledende lyd/video komprimeringsstandard, kvaliteten på denne er omtrent som en VHS-videokassett.
- **MPEG-2**
  Mye brukt i DVD og Digital TV
- **MPEG-3**
  Opprinnelig utviklet for HDTV, men benyttes ikke lenger.
- **MPEG-4**
  Inkluderer støtte for AV-objekter, 3D-innhold, lav bitrate-koding og DRM.

Fordeler med video er at man får mer data da man har tilganger på mange bilderammer per video, og du kan for eksempel plukke ut de rammene som er best:

- Vente på en god ramme.
- Velge en ramme med mindre støy.
- Vente til motivet er nærmere kameraet [8].

3.1.4 Forskjell mellom bilder og video

Noen av forskjellene mellom stillbilder og video:

- Det viktigste problemer er at videoprosessering i sanntid gjøres mest ved videofrekvenser (for eksempel 30 bilder per sekund), det vil si en ramme på 1/30 sekund.
- Beregningskravene må senkes, og pikseloppløsningen på video er lavere enn stillbilder.
- Kvaliteten på hver enkelt ramme trenger ikke å være så høy som et tilsvarende stillbilde.
- Ingen stokastisk støyreduksjon, siden dette vurderes å være adressert av den midlertidige middelverdi [8].

3.2 Maskinlæring

Maskinlæring er et fagfelt underlagt kunstig intelligens. Det er tverrfaglig med bidrag fra informatikken, matematikken og statistikken, men henter også inspirasjon fra bl.a. biologien og psykologien. Målet med maskinlæring er å gi datamaskiner mulighet til å lære, og ta beslutninger basert på data, på samme måte som vi mennesker tar beslutninger basert på det vi opplever. Akkurat som mennesker må lære utvikles algoritmer som gjør datamaskinen i stand til å lære. Disse algoritmene tar inn store mengder data, og basert på disse, vil den etterhvert bli i stand til å ta beslutninger ut fra ny data.

Teknikker innen maskinlæring:
Det er hovedsaklig tre teknikker som brukes innen maskinlæring;

Forsterkende læring

I forsterkende læring gis maskinen ett mål eller én oppgave, og gjennom utallige forsøk med prøving og feiling skal den komme nærmere en optimal løsning. Denne teknikken henter inspirasjon fra psykologiens behaviorisme, hvor tidligere valg påvirker de neste [9]. Dette er en populær teknikk for å løse komplekse logiske spill som f.eks. sjakk. En slik teknikk ble brukt av Googles DeepMind-prosjekt; AlphaZero som slo verdemnsmestre i både sjakk, shogi og Go [10]. En mulig forklaring på dette kan være at maskinen kun gis det overordnede målet med spillet; å vinne, i motsetning til mennesker som gjerne trenes ved å få vite hvordan man vinner. Dermed kan en slik maskin finne løsninger som et menneske muligens ikke ville vurdert fordi det er for risikabelt. Dette er derimot ikke en teknikk som er spesielt godt egnet til vårt prosjekt da det ikke er mange kompliserte avgjørelser som skal tas for å avgjøre om fisken er kultivert eller vill.

Ikke-veiledet læring

Ikke-veiledet læring er en teknikk hvor alle beslutninger overgis til algoritmen. Data som skal analyseres mates inn og algoritmen gir et resultat ut ut fra hva slags sammenhenger den finner i dataene. Dette er dermed en god teknikk for å finne grupperinger i data, og er populær til bl.a. å segmentere kundegrupper og sortere bilder i et galleri basert på f.eks. hva som er på bildet eller hvor de er tatt. Man sier dermed ingenting om hva disse dataene er, eller hva algoritmen skal se etter. På grunn av dette er heller ikke ikke-veiledet læring et godt alternativ for oss hvor vi har klare definisjoner på hva dataene inneholder, og hva vi ønsker at algoritmen skal gi som resultat.

Veiledet læring

Den tredje teknikken er veiledet læring. Her har vi definert hva vi ønsker at resultatet skal være. Dataene vi gir algoritmen er på forhånd sortert etter svaret vi ønsker den skal gi basert på ny data. Denne teknikken kan sammenliknes med hvordan man lærer barn forskjellen på objekter. F.eks. ved å vise en banan og si "Dette er en banan", og vise et eple og si "Dette er et eple". Slik vil et barn lære at en gul, avlang og buet frukt er en banan, og at en rund, rød, gul eller grønn frukt med glatt skall er et eple. Hvis du senere viser en ny frukt, f.eks. en ananas vil et barn mest sannsynlig spørre "Hva er det?", og skjønne at den ikke er en banan eller et eple. Her er derimot en slik algoritme mye svakere. Hvis den er lært til å se etter epler og bananer vil alt den får av data klassifiseres som enten eple eller banan. Svakheten er da at hver gang en ny klasse innføres må algoritmen trenes på nytt med ny data, og all tidligere data. Vi visste på

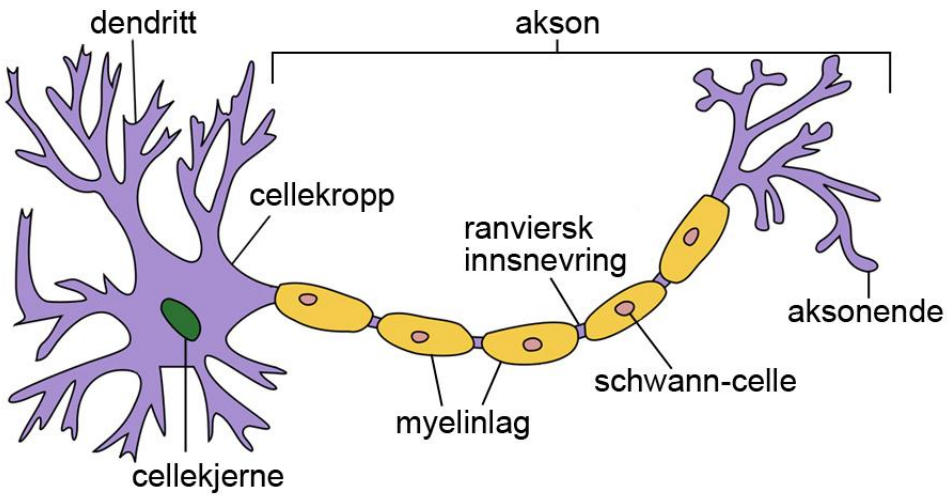
forhånd at vår data bestod av kun villfisk og settefisk, og vi skjønte raskt at dette var en god teknikk for å løse vårt problem. Dermed rettet vi fokuset mot denne teknikken og hvilke metoder innenfor dette vi kunne bruke.

3.3 Dyp læring

Det er mange måter man kunne løst vår oppgave på, kanskje hadde man ikke trengt å bruke maskinlæring i det hele tatt. Vi hadde ingen krav til hvordan den skulle løses, så det var i stor grad opp til oss å velge. Vi hadde begge interesse for, men ingen erfaring med emnet. Denne oppgaven så vi på som en gylden mulighet til å lære om dette, samtidig som vi får håndfast erfaring og gode veiledere.

3.3.1 Nevralt nettverk

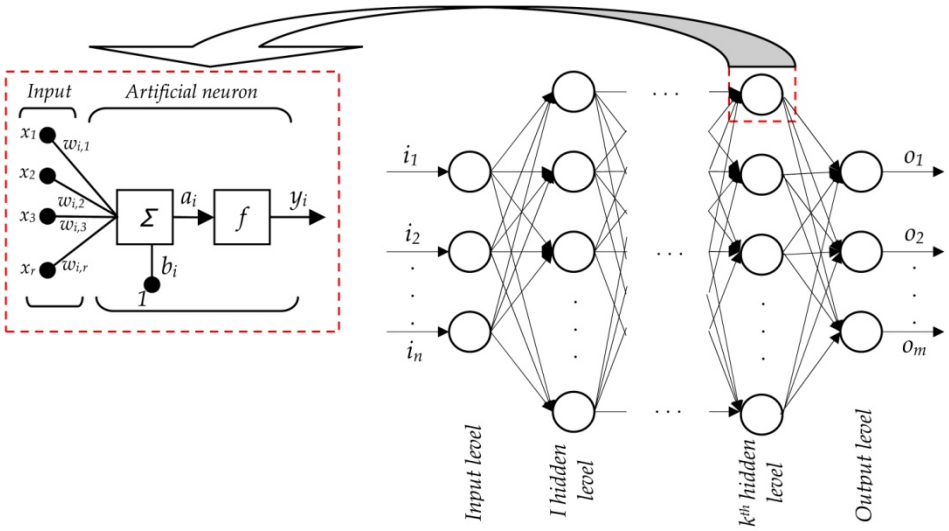
Nervesystemet hos mennesker består av millioner av nerveceller eller nevroner. En nervecelle består av en cellekropp med cellekjerne, dendritter og aksoner som kan sees på figur 6. Disse er bundet sammen av synapser, i et nettverk som sender elektriske signaler til hverandre. Signalene mottas av en celle som aktiveres, og ut fra dette sender et signal til én eller flere andre celler. Dette skjer helt til målcellen nås, og resultatet kan være at hjernen gjenkjenner en hund basert på det man ser.



Figur 6: Nervecelle [11]

Et kunstig nevralt nettverk er bygget for å være mest mulig likt et biologisk nervesystem, og er inspirert av dyrenes synssenter. Kunstige nerveceller eller noder settes sammen lagvis, disse tar input fra én eller flere nevroner fra forrige lag, og basert på dette sender sin output til et nytt nevron. Lagene danner nettverket slik det vises på figur 7. I slike nettverk kan antall lag og antall noder i hvert lag variere, men to ting vil alltid være de samme for samme problem: Input- og output-laget. Lagene i mellom disse kalles som regel skjulte lag, og nettverk med flere enn ett skjult lag omtales gjerne som dype nevrale nettverk.
Input-laget avhenger av hva man skal trene på, eller klassifisere. I vårt tilfelle skal nett-

verket ta inn bilder. Bilder består som nevnt i kapittel 3 av antall piksler i bredden, antall piksler i høyden og antall kanaler. Input-laget vil da ha like mange noder som produktet av disse tre verdiene. På motsatt side har vi output-laget. Det er her resultatet kommer ut, og vil dermed bestå av like mange noder som antall klasser man ønsker å klassifisere. F.eks. to klasser; settefisk og villfisk, vil gi kun to noder i output-laget.

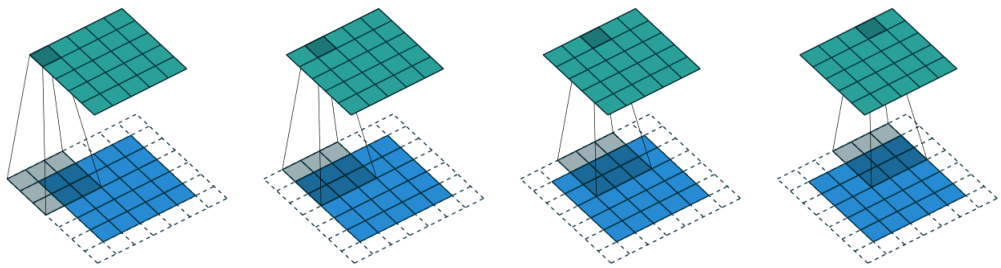


Figur 7: Kunstig nevralt nettverk [12]

3.3.2 Konvolusjonelt nevralt nettverk

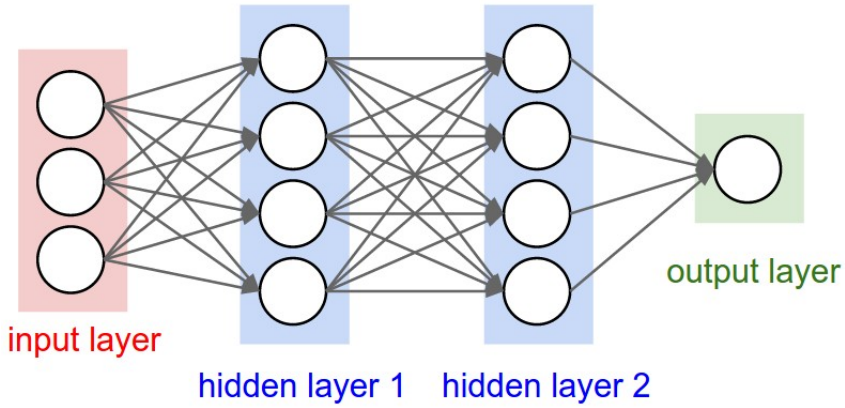
Konvolusjonelt nevralt nettverk (KNN) er blitt de mest populære for å løse oppgaver relatert til datasyn. Dette fordi de lenge har gitt de klart beste resultatene i slike applikasjoner [13]. Navnet stammer fra de skjulte lagene i et slikt nettverk som inneholder minst ett lag som er konvolusjonelt. Lagene i slike nettverk kan bl.a. bestå av:

• Konvolusjonelle lag (convolutional layer) Konvolusjon betyr bretting og er hentet fra bildebehandling i datasyn. Det går ut på at et filter, typisk en 3 X 3-matrise, kjøres over hver output fra nodene i forrige lag. Produktene av output i denne noden, og alle rundt i samme størrelse som filteret, og korresponderende node i filtermatrisen summeres og gir resultatet i noden. Eksempel på dette kan sees på figur 8. På den måten "brettes" de rundtliggende verdiene inn i en node, derav navnet.



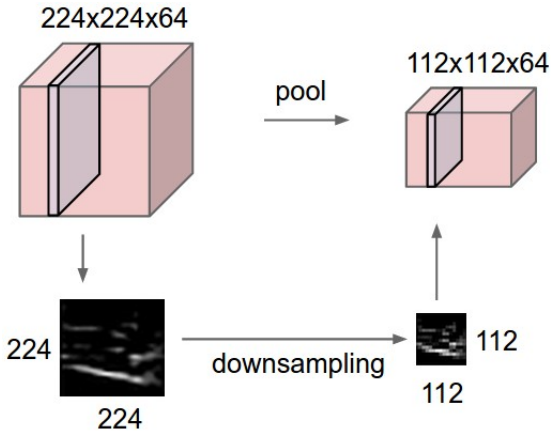
Figur 8: Konvolusjon [13]

• Fullt sammenkoblede lag (fully connected layer) I fullt sammenkoblede lag er hver node i laget koblet til hver node i neste lag, og et banalt eksempel kan sees på figur 9. Tradisjonelle nevrale nettverk var bygget av primært slike lag, og man så at dette førte til overfitting [14]. Dette fører også til veldig mange parametre gjør algoritmen mer ressurskrevende.



Figur 9: Fullt sammenkoblede lag [15]

• Samlende lag (pooling layer) Disse lagene brukes til å nedskalere antall parametre bl.a. for å forhindre overfitting. Dette kan gjøres på flere måter, men den mest brukte i knn er max-pool. Denne metoden tar et antall nærliggende verdier og gir for denne blokken en enkelt verdi ut. I max-pool er dette den høyste verdien i hver blokk. Et eksempel på dette kan sees på figur pooling hvor blokker på 2 X 2 reduseres til én enkelt verdi, noe som reduserer det totale antallet med 4 ganger.



Figur 10: Max-pooling [15]

3.3.3 Oveføringslæring

Overføringslæring er en maskinlæringsmetode der en modell utviklet for én oppgave blir gjenbrukt som utgangspunkt for en modell på en annen oppgave.
Det er en populær tilnærming i dyp læring der ferdigtrente modeller brukes som utgangspunkt for datasyn og oppgaver for naturlig språkbehandling. Overføringslæring

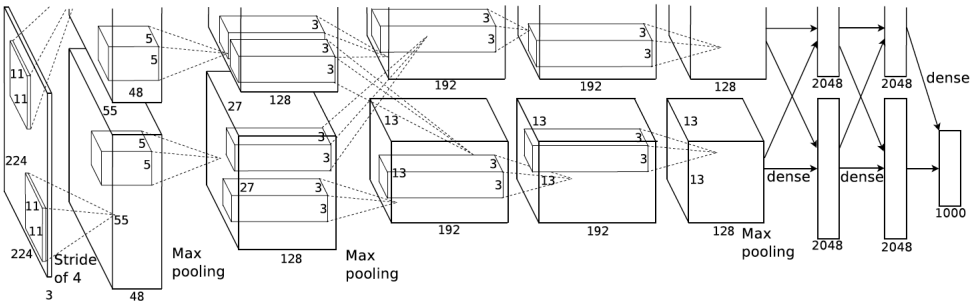
brukes grunnet de enorme ressursene som trengs for å trene dype læringsmodeller, eller de store og utfordrende datasettene som dype læringsmodeller er opplært på. Teknikken virker bare i dyp læring dersom modellfunksjonen fra den første oppgaven er generell. Det vil si f.eks. når vi overfører læring, trener vi først et basis nettverk på et basis datsett og en oppgave, deretter tilbakestiller vi de lærte funksjonene, eller overfører dem til en annen destinasjon hvor nettverket vil bli trent på et annet datasett og oppgave. Denne prosessen vil ha en tendens til å fungere hvis funksjonene er generelle, noe som betyr at de passer til både basis- og måloppgaver, i stedet for spesifikk for basisoppgaven [16].

3.4 Modeller innen dyp læring

ImageNet var benyttet som treningsdata i de ferdigtrente nettverkene vi har benyttet oss av. ImageNet er et datasett med over 15 millioner høyoppløsningsbilder som er kategorisert og består av 22.000 forskjellige kategorier [17].

3.4.1 AlexNet

AlexNet er et KNN som er en klasse av dype nevrale nettverk for bilde klassifisering. Det er et ferdigtrent nettverk som er tilgjengelig i Matlab. AlexNet er kjent for å vunnet konkurransen ImageNet Large Scale Visual Recognition Challenge (ILSVRC) i 2012. Treningsdataene til nettverket består av 1.2 millioner ImageNet-bilder og 1000 kategorier kalt "mini-batch". AlexNet består av 8 lag, de første fem er "convolutional" og de tre siste er "fully connected" lag [18].



Figur 11: Arkitektur AlexNet - to identsike sett med lag [18]

Noen høydepunkter fra arkitekturen på figur 11:

• Mer data og større modell med 7 skjulte lag, 650.000 enheter og 60 millioner parametre.

• Benytter seg av ReLu funksjon istedenfor Sigmoid eller Tanh funksjoner. Yter 5 ganger raskere med samme nøyaktighet [19].

• Den benytter seg av Dropout istedenfor regularisering for å håndtere overfitting, men treningstiden dobles med frafallshastighet på 0,5.

3.4.2 GoogLeNet

GoogLeNet er et KNN som trener på enten ImageNet eller Places 365 datasett. GoogLeNet er kjent for å vunnet konkurransen ILSVRC i 2014. GoogLeNet består av 22 lag, mot AlexNets 8, noe som gjør det mye mer ressurskrevende å trene [18] [10].

3.4.3 Valget av ferdigtrent modell

For valg av ferdigtrent modell la vi vekt på hvor nøyaktig nettverket blir og hvor lang tid det tar å trene nettverket. Da vi ikke har så god tid og ønsker forsøke mest mulig er det viktig at vi ikke bruker for lang tid på selve treningen.

Blant modellene AlexNet og GoogLeNet, ble det **AlexNet** vi havnet på.
Noen punkter som var med på avgjørelsen vår:

• AlexNet trener raskere enn GoogLeNet, som gjør at vi får testet forskjellige teknikker hyppigere en å måtte vente lenge på hver trening av nettverk.

• En tilsvarende oppgave har blitt gjennomført før med bruk av overføringslæring, og der ble det påvist omtrent samme nøyaktighet når man trente med samme parameter, men AlexNet brukte halvparten av tiden GoogLeNet brukte [20]. Vi testet begge modellene opp mot hverandre med samme treningsparameter og datasett. Der endte vi opp med at AlexNet fikk bedre nøyaktighet og brukte mindre tid enn GoogLeNet.

3.5 Læringsalgoritmer

I Matlab har vi tilgjengelig tre forskjellige læringsalgoritmer. Følgende læringsalgoritmer finnes tilgjengelig:

• SGD

• Adam

• RMSProp

• SGDM

3.5.1 Stochastic gradient descent algorithms

Stochastic Gradient Descent (SGD) er en iterativ metode for optimalisering av en differensierbar objektiv funksjon.Den blir kalt for stokastisk fordi prøvene fra treningssettet velges tilfeldig (eller blandes) istedenfor den rekkefølgen de vises i treningssettet [21].

3.5.2 Adam

Adaptive moment estimation (Adam) er en optimaliseringsalgoritme som kan brukes istedenfor den klassiske SGD for å oppdatere nettverksvekter basert på iterativ treningsdata [21].
Noen fordeler ved bruk av Adam:

• Metoden er enkel å implementere

• Har lite minnekrav

• Effektiv mot oppgaver med mye bildestøy

Hvordan fungerer Adam?
Adam er en kombinasjon av fordelene til to andre utvidelser av SGD.

• Adaptive Gradient Algorithm
(AdaGrad) som opprettholder en per-parameter læringsfrekvens som forbedrer ytelsen på problemer med få gradienter (for eksempel naturlig språk- og datasynproblemer). Denne er ikke tilgjengelig i Matlab.

• Root Mean Square Propagation
(RMSProp) som også opprettholder per-parameter læringsfrekvensen som er tilpasset basert på gjennomsnittet av nylige størrelser av gradienter for vekten (for eksempel hvor raskt den endrer seg) [21].

3.5.3 RMSProp

Root Mean Square Propagation (RMSprop) er en SGD-basert algoritme som kombinerer metodene Adaptive Gradient Algorithm (AdaGrad) og AdaDelta. Det er en god, rask og populær algoritme som mange foretrekker innen dyp læring [21].

• Adagrad
Adagrad er en SGD-algoritme som endrer skalaen til gradienten av hver parameter, hver parameter får en uavhengig læringsfrekvens [21].

• Adadelta
Adadelta er en mer robust utvidelse av Adagrad, den tilpasser seg læringsfrekvensen basert på et bevegelig område med gradientoppdateringer istedenfor å samle på alle tidligere gradienter [21]. Denne er ikke tilgjengelig i Matlab.

3.5.4 SGDM

Stochastic Gradient Descent with Momentum (SGDM) bruker en konstant læringsfrekvens for alle parametre mens optimaliseringsalgoritmene forsøker å forbedre treningsytelsen ved å benytte forskjellig læringsfrekvens for forskjellige parametre som automatisk kan tilpasse seg tapsfunksjonen for å bli optimalisert [21].

4 Datasett og preprosessering

Dette kapittelet vil introdusere hvilke verktøy og språk vi har benyttet, samt vise frem hvordan datasettet vi har fått ser ut. Vi vil også gå inn på forskjellige teknikker vi har benyttet ved preprosessering av bilder slik at man får et innblikk i hva vi har arbeidet med.

4.1 Valg av utviklingsmiljø og språk

Valget av programmeringsspråk stod mellom Python og Matlab. De er begge populære innen datasyn og maskinlæring, hvor begge har et stort utvalg pakker og verktøykasser som kan legges til. En stor forskjell er derimot prisen for å bruke de; Python er open-source og dermed gratis i bruk, det samme er også pakkene man legger til som f.eks. NumPy (for matematiske beregninger og matriser) og OpenCV (for datasyn og bildebehandling). Matlab er derimot relativt dyrt, spesielt for enkeltpersoner. Lisens for én person med et par verktøykasser koster fort titusenvis av kroner. Heldigvis for oss stiller NTNU med akademisk lisens for studenter. Dermed forhørte vi oss med veileder som guidet oss mot Matlab. Vi endte opp med å gjøre det meste av bildebehandlingen i Python, mens maskinlæringen og utviklingen av applikasjonen ble gjort i Matlab. Dette valget gjorde vi på grunnlag av den gode dokumentasjonen som finnes for Matlab, både på deres egne hjelpesider og i diverse åpne fora. Alle funksjoner, og tildels temaer har en egen side med forklaringer og ikke minst eksempler man kan kjøre ved å klikke én knapp, og endre som man vil.

4.2 Datasett

I denne seksjonen vil vi gå igjennom hvordan vi har bearbeidet datasettet vårt.
Vi har mottatt datasettet vårt fra NINA, der fikk vi tildelt totalt 204 videosnutter. Litt informasjon rundt datasettet:

- 101 videosnutter av kultivert ørret
- 103 videosnutter av villfisk
- Varigheten på hver videosnutt er 24 sekunder
- Oppløsning 320 x 240

Fra datasettet har vi hentet ut de beste bilderammene som vi har bearbeidet i analysen vår. Mer informasjon rundt uthenting av bilder vil du se i seksjonen under.

Datasettet deler vi så inn i tre forskjellige klasser/mapper:

Figur 12: Klassestruktur

• Ingenfisk
Her ligger alle bildene hvor det ikke er fisk.

• Settefisk
Her ligger alle bildene av settefisk, hvor "fettfinnen" er klipt vekk.

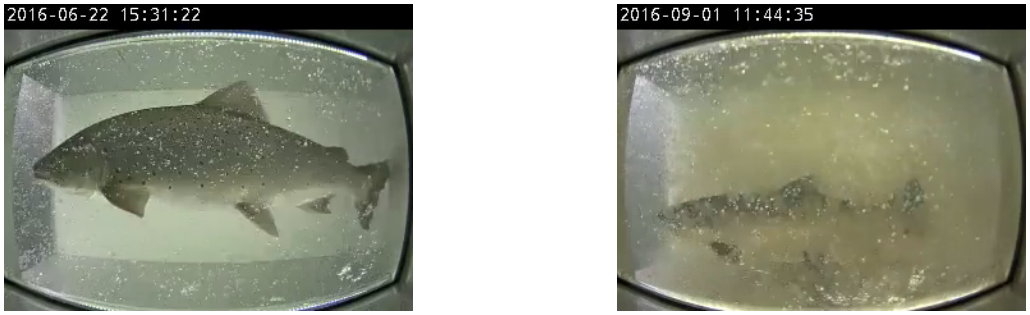
• Villfisk
Her ligger alle bildene av villfisk, hvor "fettfinnen" er intakt.

4.2.1 Konvertering fra video til bilder

Det første vi gjorde med datasettet var å konverte fra video til bilder for å senere benytte disse i treningen. Dette gjorde vi først i Python ved bruk av OpenCV-biblioteket. Biblioteket har funksjonene VideoCapture() som leser en video, read() som leser en og en bilderamme. Dermed var det bare å lagre disse bildene med et passende navn ved bruk av imwrite() som skriver filen til disk. Da vi konverterte fra videosnutter til bilderammer fikk vi rundt 600 bilder per video. Vi fikk som regel mellom 30-40 bilder der vi så fisken, av de 600. Resten var tomme, altså at det ikke var noen fisk i bildet. Da vi hadde konvertert alle de 204 videosnuttene fikk vi ca. 170.000 bilder.

Kvalitet i video

Kvaliteten i videoen er varierende, enkelte videoer har god kvalitet og belysning, mens andre har mye forstyrrelser som for eksempel luftbobler. Dette er elementer som kan være med på å forstyrre slik at fisken blir klassifisert feil. Nedenfor har vi to eksempler fra forskjellige videosnutter som viser et bilde uten noen særlig forstyrrelser, og et med mye forstyrrelser.



(a) (b)

Figur 13: Eksempel på bilderamme

På figur 13a har vi et godt eksempel på et bilde uten forstyrrelser, her ser man hele fisken i sin helhet og "fettfinnen", dette er blant de beste bildene vi har tilgjengelig. På

figur 13b, ser vi et dårligere eksempel da det er en annen belysning og mye forstyrrelser i form av luftbobler, som gjør at vi ikke kan se om det er en settefisk eller villfisk.

4.2.2 Filtrering av datasett

Filtrering av datasett vil si at vi sorterer bort de dårligste bildene og beholder de beste bildene fra videosnuttene.
Vi forsøkte å automatisere prosessen med å filtrere ut bildene, men det viste seg å være trøblete. Istedenfor at vi forsøkte å lese oss opp og se på forskjellige teknikker på internett bestemte vi oss for å gjøre dette manuelt. Dette sparte vi mye tid på kontra å finne en metode som automatiserte dette da datasettet ikke var så stort.

Av 170.000 bilder satt vi igjen med rundt 14.000 bilder fordelt på settefisk/villfisk. Vi hadde ujevnt datasett i starten, så vi måtte kutte ned og velge kun de beste bildene fra settefisk/villfisk og endte opp med 5.000 bilder av hver. Vi fant ut etterhvert som vi hadde testet at vi måtte tilføye en ekstra klasse "ingenfisk". Grunnen til det kommer vi tilbake til i kapittel 5. Da filtrerte vi ut de 25 første bildene av hver videosnutt som ikke inneholdt fisk. Totalt med tre klasser hadde vi nå 15.000 bilder, 5.000 i hver kategori, i vårt datasett.

4.2.3 Filformat

Filformatet vi velger når vi konverterer fra video til bilde er viktig å tenke på. Første gang vi konverterte valgte vi .jpg som filformat, men vi hadde ikke tenkt på hva de forskjellige egenskapene til filformatene var. Her fikk vi god tilbakemelding fra veilederen vår at vi burde benytte .png filtype. Da vi gjorde dette ble det datasettet betydelig større og vi så ikke noen stor forskjell, men det kan være viktige detaljer på bilde som blir borte.

• JPG-format
JPG bruker en komprimeringsmetode med kvalitetstap. Bildestørrelsen blir mindre, men viktige detaljer kan bli borte i komprimeringsprosessen. Brukes som regel på digitale fotografier.

• PNG-format
PNG er et ikke-komprimerende filformat, det vil si at det ikke gir kvalitetstap. PNG blir ofte brukt til linjetegninger, tekst og grafikk med liten filstørrelse [22].

(a) PNG-format, størrelse = 116kB

(b) JPG-format, størrelse = 30,5kB

Figur 14: PNG format vs JPG format

Her ser vi to like bilder, men med forskjellig filformat og størrelse. PNG-formatet er tapsfritt i forhold til videosnutten, mens JPG-formatet er komprimert og kan da ha mistet viktige detaljer i bildet, selv om man ikke kan se det ut ifra en slik visuell sammenligning.

4.3 Bildebehandling

I denne seksjonen vil vi snakke om bildebehandling, altså hvordan vi kan kan forstørre og/eller forbedre datasettet vårt. Vi undersøker om det blir bedre resultat om vi benytter oss av forskjellige fargekanaler og kontrastforbedringer.

4.3.1 Preprosesseringsteknikker

Datasettet vårt skal brukes til maskinlæring, som vi går gjennom i detalj i kapittel 5. For å lage en god modell er det viktig at man har nok data å trene på [23]. Desto mer man trener, desto bedre blir man i det man trener på, hvis man trener riktig vel og merke. Mer om dette i neste kapittel. For å oppnå en tilfredsstillende mengde data så vi på ulike metoder for augmentering eller utvidelse av datasettet. Forskjellige teknikker vi har benyttet oss av:

CLAHE

Contrast-limited adaptive histogram equalization (CLAHE) brukes for å se om kontrastforbredningen vil hjelpe nettverket.

(a) Uten CLAHE

(b) Med CLAHE

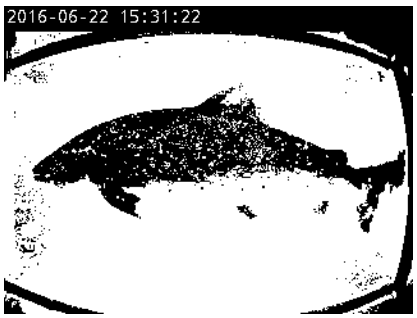
Figur 15: Uten og med CLAHE

På figur 15 ser vi et bilde med og uten CLAHE. Denne teknikken brukes i rapporten under kapittel 5. Denne teknikken brukes primært på gråskalabilder da den påføres på én enkelt kanal, og har tidligere vist gode resultater ved bruk på undervannsbilder [24]. CLAHE virker ved at man påfører flere histogramutligning på bildematrisa. Dermed blir skillene mellom fargene i bildet mer uthevet, slik at konturene til objektene kommer tydeligere frem. Vi hadde fargebilder så vi måtte finne en litt annen løsning. Dette gjorde vi i Python. Det vi måtte gjøre var å konverte fra RGB til LAB og splitte kanalene. RGB og LAB er begge fargerom, men med litt annen definisjon. "L" er lysheten i bildet, "A" er fargene magenta (120) til grønt (-120), og "B" er fargene gult (120) til blått (-120). Etter konverteringen påførte vi CLAHE på "L"-kanalen ved bruk av funksjonen createCLAHE() hvor vi spesifiserte CLAHE-innstillingene, (clipLimit=2.0, tileGridSize=(4,4)). Til slutt påførte vi innstillingene på "L"-kanalen med clahe.apply(), slo sammen kanalene og lagret bildet.

Binært bilde

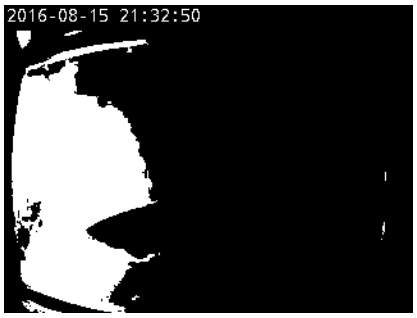
Brukes for å finne et punkt av interesse, i vår tilfelle "fettfinnen". For å konvertere bilde til binært må vi først konvertere til gråskala. Når dette er gjort konverterer vi videre til binært ved å bruke funksjonen "imbinarize" i Matlab. Dette fungerer ved at man setter en grenseverdi, et tall mellom (0-255) for all pikslene i bildet. De som er under grensa blir satt til 0 (sort), og de over til 255 (hvitt).

(a) Villfisk - fargebilde



(b) Villfisk - binært bilde

(c) Settefisk - fargebilde



(d) Settefisk - binært bilde

Figur 16: Originale og binære bilder

De to første figurene på figur 16, 16a og 16b, ser vi samme bilde av en villfisk, der vi har ett originalt bilde og en med binære kontraster. I dette tilfelle ser vi at vi får "forsterket" fettfinnen men mister litt av undersiden av fisken. Om vi ser på figur 16d som er binære kontraster av et bilde med en settefisk ser vi at vi ikke får noe utbytte da

hele fisken blir omtrent borte. Vi valgte derfor ikke gå videre med denne teknikken da bildene våre har varierende lysforhold.

Gråskala

RGB-bilder inneholder mye datainformasjon som kanskje ikke er nødvendig for bildebehandling, derfor har vi konvertert bilder til gråtskala. For å konvertere til gråskala tar vi gjennomsnittet av de tre fargekanalene RGB og setter sammen bildet til et gråskalabilde. Dette gjøres automatisk i python ved å angi imread_grayscale() ved innlasting av bildet.

(a) Original

(b) Gråtone

Figur 17: Farge og gråtone

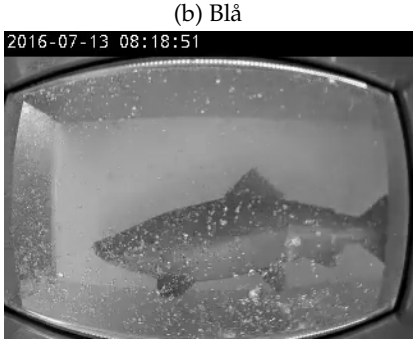
Fargekanaler

Brukes for å se om forskjellige fargekanaler kan hjelpe oss med å identifisere kanter eller andre egenskaper i bildet. Dette gjøres i Python ved å laste inn bildet i farger, og deretter splitte opp de tre fargekanalene til hvert sitt en-kanalsbilde.

(a) Vanlig

(b) Blå

(c) Grønn



(d) Rød

Figur 18: Fargekanaler

Her på figur 18 har vi fire bilder med forskjellige fargekanaler. Det er veldig vanskelig å se om det finnes noen forskjell fra figur 17b, fra gråskala, til de tre andre figurene 18b, 18c og 18d. Vi har etterspurt oppdragsgiver om kameraspesifikasjoner, men oppdragsgiver hadde ingen spesifikke opplysninger rundt kameraet som blir brukt.

Uskarphet

Uskarphet kan legges til et bilde for å redusere støy og detaljer. Det gjøres ved at en Gauss-funksjon legges til originalbildet. Dette er det samme som å konvolvere et bilde med en Gauss-funksjon, som vi kommer tilbake til i kapittel 5. Denne funksjonen "glatter" ut verdiene til pikslene i et område, slik at de høyeste verdiene senkes, som i et lavpass-filter. For å legge til Gauss-funksjonen benyttet vi "cv2.GaussianBlur()" i Python.

(a) Original

(b) Uskarp

Figur 19: Originalt og uskarpt bilde

Figur 19 viser bildet før og etter uskarphet er lagt til. I og med at vi jobber med undervannsbilder er mange av bildene såpass uskarpe originalt at vi ikke fant dette hensiktsmessig å bruke.

Beskjæring og skalering

Å beskjære et bilde vil si å fjerne en del av det. Fordelen med dette er at man kan velge forskjellige utsnitt fra ett enkelt bilde, og på den måten få flere bilder ut av ett. Bildematrisa for dette området vil være eksakt den samme som området man skar ut fra originalbildet. Det vil si at hvis man har to like bilder på papir, i samme størrelse, og klipper det ene i to, så vil denne halvdelen passe over det samme området på det bildet som er helt. Dette gjorde vi i Python ved å sette koordinatene for utsnittet vi ønsket, f.eks. x=20, y=0, h=210 og w=320 og påføre disse på bildet. Disse koordinatene ble brukt for å lage bildet på figur 20b under.

En lignende metode er skalering, hvor man også velger et utsnitt fra originalbildet. For å skalere et bilde benytter man funksjonen "Image.resize" i Python hvor man må legge inn ønsket piksler, som f.eks. 300 x 210, som ble gjort med bildet på figur 20c under. Forskjellen på disse er at ved skalering beholder man den originale bildestørrelsen, men bildematrisa for det originale og det skalerte bildet vil være forskjellig. Dette krever med andre ord en prosessering i etterkant for å skalere eller "zoome inn" dette området slik at man oppnår samme bildestørrelse.

(a) Original

(b) Beskåret

(c) Skalert

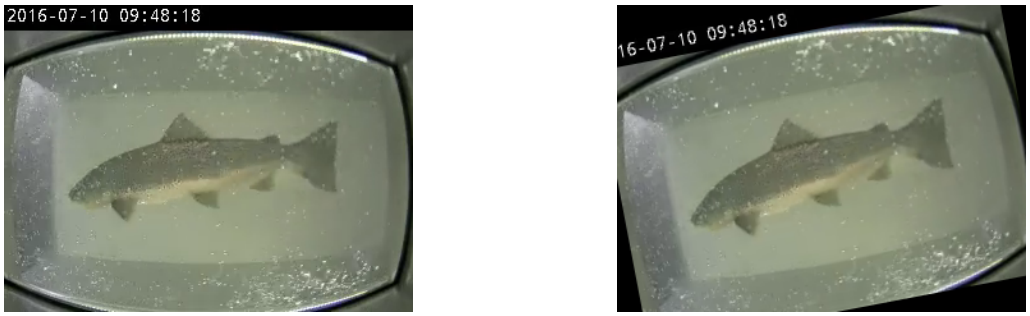
Figur 20: Originalt, beskåret og skalert bilde

På figur 20 vises forskjellen på disse metodene. Bildet på figur 20b er beskåret og er nå av størrelse 300 X 210 piksler, mens bildet på figur 20c viser det samme området, men er altså skalert opp til samme størrelse som det originale, 320 X 240. Ulempen med begge disse metodene i vårt datasett er at objektet (fisken) ikke befinner seg på samme

sted i alle bildene. Dermed er det en stor utfordring å velge et utsnitt hvor vi er garantert at fettfinnen er synlig fra alle 20.000 bildene i datasettet som inneholder fisk. I tillegg må man som nevnt utføre en prosessering ved skalering noe som kan forringe kvaliteten og på den måten gi et dårligere resultat.

Rotasjon

Rotasjon er rett og slett en dreining av bildet om et rotasjonspunkt, som regel midtpunktet. Her er det mulig å variere hvor mange grader bildet skal roteres, og hver rotasjon vil gi en helt annen bildematrise. Det som skjer er en translasjon av hvert punkt (piksel) i (x,y)-planet til sin nye plass. Verdien i denne pikselen skrives altså til denne nye plassen. Dette gjorde vi også i Python med funksjonen iaa.Affine(rotate=(-25,-25)) fra imgaug-biblioteket.



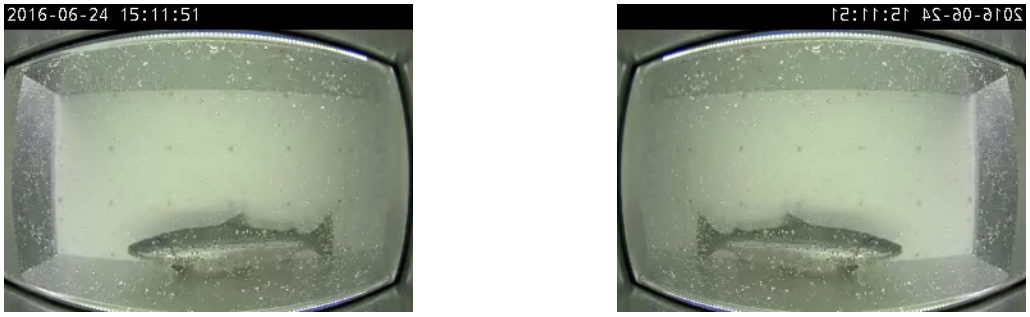
(a) Original (b) Rotert

Figur 21: Originalt og rotert bilde

Figur 21 viser et eksempel på en slik rotasjon hvor bildet er rotert 25 grader mot klokka. Som man ser vil man ved de fleste rotasjoner hvor den originale bildestørrelsen opprettholdes (ikke 90, 180, 270 grader ved kvadritiske bilder, eller 180 grader på rektangulære bilder) få områder hvor det ikke finnes pikselverdier. På samme måte vil man få like store områder fra det originale bildet som havner utenfor rammen. I vårt tilfelle settes verdiene innenfor til 0, som gir sorte områder, mens pikslene som havner utenfor klippes ut av bildet.

Vending

Den siste teknikken vi så på var vending av bilder. Bildene kan vendes horisontalt og vertikalt. Ved en horisontal vending transleres hver enkelt kolonne i bildematrisa til korresponderende plass på motsatt side, slik at objekter som er vendt mot venstre blir vendt mot høyre og motsatt. På samme måte ved vertikal vending transleres hver enkelt rad. Dette ble gjort med funksjonen iaa.Fliplr(1) fra imgaug-biblioteket i Python.



(a) Uten horisontal flip

(b) Med horisontal flip

Figur 22: Med og uten horisontal flip

4.3.2 Valg av teknikker

Av teknikkene vi gjennomgikk i dette kapittelet endte vi opp med benytte CLAHE og vending av bilder.
Hovedårsaken til at vi gikk for CLAHE var på grunnlag av en artikkel hvor det ble gjort lignende klassifisering hvor de fikk over 99% ved bruk av CLAHE [20].
Når det kommer til vendig av bilder tok vi ikke dette i bruk før i slutten av prosjektperioden. Vi ønsket å trene på et større datasett, og med vending av bilder ga det oss et dobbelt så stort datasett som vi har i utgangspunktet. For resultater gjort med teknikkene kan man se det i kapittel 5.

5 Implementering og realisering

I dette kapittelet tar vi for oss trening av maskinlæringsmodeller med to og tre klasser, utvikling av applikasjonen og en gjennomgang av maskinvaren vi har brukt.

5.1 Maskinlæring - trening

Versjonen vi benytter for treninger og klassifisering i Matlab er versjonen:"Matlab R2018b". I tillegg er det nødvendig med "verktøykassene" "Deep Learning Toolbox 12.0" og "Statistics and Machine Learning Toolbox 11.4"
For trening av nettverk med GPU trenger man "Parallell Computing Toolbox".

5.1.1 Trening av KNN og ECOC med to klasser

Trening av KNN

Når vi hadde gjort vår undersøkelse av læringsalgoritmer ønsket vi å teste dette ut i praksis. På Matlabs hjelpesider fant vi en god artikkel med et eksempel på Overføringslæring som vi kunne tilpasse [25]. Dette scriptet benyttet AlexNet, et ferdigtrent nettverk som lå tilgjengelig i Matlab. Vi prøvde først å kjøre eksempelet med tilhørende eksempel-data, da det fungerte greit startet jobben med å tilpasse til vårt bruk. Matlab bruker objekter kalt "imageDataStore" for å ta inn bilder i store volum. Vi fant ut hvordan vi kunne lage en slik av vårt datasett der vi benyttet to klasser, settefisk og villfisk. I hver kategori var det 5.000 bilder, totalt 10.000. Nettverket ble trent med et grafikkort (GeForce GTX 1080Ti) på følgende innstillinger.

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',10, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',1e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Listing 5.1: Treningsinnstillinger

- MinBatchSize

Batchsstørrelsen er en hyperparameter som definerer antall prøver som skal gjennomføres før man oppdaterer de interne modellparametrene. Et treningssett kan deles inn i en eller flere batcher. En miniBatch er en delmengde av treningssettet som brukes for å evaluere gradienten av tapsfunksjonen og oppdatere vekter.

- Epochs

En epoke betyr at hver prøve i trenings-datasett har hatt mulighet til å oppdatere de interne modellparametrene. En epoke består av en eller flere "batcher"

• InitialLearnRate
Mengden av vektene som oppdateres under trening blir referert til som læringsfrekvensen. Det er en konfigurerbar hyperparameter som brukes i trening av nevrale nettverk som har en liten positiv verdi, mellom 0.0 og 1.0.

Nøyaktighet i %	SGDM	Adam	RMSProp
Uten CLAHE	94.06%	91.33%	81.20%
Med CLAHE	91.46%	93.07%	86.26%

Tabell 1: Nøyaktighet av læringsalgoritmer

I tabell 1, har vi kun kjørt treningen en gang på hver av læringsalgoritmene. Ut ifra denne tabellen ser vi at vi kan fjerne RMSProp, da denne gir lav nøyaktighet i forhold til SGDM og Adam. Videre søkte vi etter litteratur om hvor mange treninger vi burde kjøre og ta gjennomsnitten av dette. Vi fant ikke noe gode innspill, da det er veldig varierende og ulike faktorer som er avgjørende.

Results	
Validation accuracy:	94.43%
Training finished:	Reached final iteration
Training Time	
Start time:	21-Mar-2019 23:07:42
Elapsed time:	201 min 56 sec
Training Cycle	
Epoch:	6 of 6
Iteration:	4200 of 4200
Iterations per epoch:	700
Maximum iterations:	4200
Validation	
Frequency:	3 iterations
Patience:	Inf
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

(a) SGDM-trening

Results	
Validation accuracy:	91.33%
Training finished:	Reached final iteration
Training Time	
Start time:	22-Mar-2019 09:02:31
Elapsed time:	307 min 10 sec
Training Cycle	
Epoch:	6 of 6
Iteration:	4200 of 4200
Iterations per epoch:	700
Maximum iterations:	4200
Validation	
Frequency:	3 iterations
Patience:	Inf
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

(b) Adam-trening

Figur 23: Treningsresultat av SGDM og Adam.

Som vi ser på figur 23a, tar det rundt 200 minutter for å trene datasettet og på figur 23b tar det rundt 300 minutter. Siden det tar rikelig med tid har vi satt av tid til å kjøre fem treninger av hver og ta gjennomsnittet av nøyaktigheten på treningene.

Nøyaktighet i %	SGDM	Adam
1	91.57%	93.13%
2	94.43%	93.57%
3	95.10%	91.97%
4	94.06%	91.33%
5	99.15%	94.57%
Gjennomsnitt	94.86%	92.91%

(a) Fem treninger uten CLAHE

Nøyaktighet i %	SGDM	Adam
1	91.46%	91.33%
2	92.23%	92.23%
3	93.77%	88.43%
4	86.60%	89.13%
5	89.23%	93.89%
Gjennomsnitt	90.65%	91.00%

(b) Fem treninger med CLAHE

Tabell 2: Trening med og uten CLAHE

Ut ifra tabell 2 ser vi i 2a at det er SGDM har det beste nøyaktigheten gjennom fem treninger, selv om det ikke er mye som skiller seg fra Adam sin nøyaktighet. I 2b ser vi derimot at Adam har den beste nøyaktigheten, men SGDM er rett rundt hjørnet. Som vi var inne på under figur 23, er det også røflig 100 minutter som skiller de to læringsalgoritmene. Det at det tar mindre tid og har bedre nøyaktighet er positivt for oss.
Som vi har vært inne på tidligere har tidsbruk og nøyaktighet vært en viktig faktor for vår videre prosess. Vi har lest oss opp på litteratur som har gjort eksperimentell bevis som viser til at adaptive metoder ikke er fordelaktig for maskinlæring, de er også usikre på hvorfor Adam-algoritmen fortsatt er populær blant maskinlæring, da de har funnet ut at det er bedre å praktisere SGD ved maskinlæring [26]. Det siste nettverket med 99.15% nøyaktighet ble trent etter vi tok i bruk horisontal vending som augmenteringsteknikk, og fikk som en følge av det dobbelt så stort datasett. Det vi leste om å øke mengden med data for å bedre resultatet viste seg å stemme [23].

Trening av ECOC-klassifiserer

Etter vi hadde trent nettverket med to klasser, kjørte vi gjennom en Error correcting ouput codes (ECOC)-klassifiserer. En ECOC-klassifiserer reduserer et klassifiseringsproblem med n-klasser til flere små binære problemer. Typisk er disse binærklassifisererene Support vector machine (SVM)'er. SVM'er har vist seg å være gode på klassifiseringsproblemer, så vi bestemte oss for å prøve ut dette i Matlab[27]. Treningen av denne skjer ved å bruke en allerede trent modell hvor egenskapene (features) som er hentet fra bildene ligger lagret, et KNN i vårt tilfelle. Deretter velger man ett lag, og dette lagets egenskaper vil bli overført til ECOC-klassifiseren. Denne kjører så gjennom treningsdataen og tilpasser seg disse egenskapene ved å bruke de som variabler i treningen. Med to klasser ender vi opp med kun én binærklassifiserer, som klassifiserer som enten settefisk eller villfisk. Fordelen med ECOC-klassifiserer er at den er veldig rask å trene da den kjører gjennom treningsdataen én gang per trening.

Nøyaktighet i %	KNN	ECOC
1	94.06%	95.15%
2	94.06%	95.18%
3	94.06%	95.19%
4	99.15%	99.88%
5	99.15%	99.90%
6	99.15%	99.92%

Tabell 3: ECOC

På tabell 3 ser vi hvilket resultat vi har fått med to forskjellige nettverk med nøyaktigheten 94.06% og 99.15%. Vi trente ECOC klassifiseringen tre ganger for få få det beste resultatet, når vi kjører gjennom klassifiseringen får vi generelt høyere nøyaktighet enn det nettverket opprinnelig har.

Oppsummering av trening med to klasser

I treningen med to klasser testet vi forskjellige læringsalgoritmer. Vi testet både RMSprop, Adam og SGDM, dette ble gjort med og uten CLAHE. Etter treninger i starten med de ulike læringsalgoritmene fant vi fort ut at RMSProp ikke gir oss tilfredsstillende nøyaktighet til at vi kan fortsette trene på den, da satt vi igjen med SGDM og ADAM. Vi fortsatte treningen med SGDM og ADAM, kjørte de 5 ganger hver med og uten CLAHE slik at vi fikk tatt en gjennomsnittlig nøyaktighet for å se om den er stabil eller om det kun er et engangstilfelle at vi får god nøyaktighet som vi kan se på tabell 2. I tabellen fikk vi bekreftet at SGDM er den som har best nøyaktighet gjennom fem treninger.
Før vi begynte å trene nettverk, hadde vi gjort våre undersøkelser hvor det har blitt gjort lignende oppgaver, og fått et innblikk i at SGDM var blant de beste læringsalgoritmene [20]. Da vi fikk bekreftet det ut ifra vår analyse og trening var det et tegn på at dette stemte også på vårt problem.
Vi prøvde oss på trening av ECOC-klassifiserer og fant ut at den fikk en god del bedre nøyaktighet enn nettverket den ble trent fra. I begynnelsen var det bare nøyaktigheten vi hadde å gå utfra, så målet ble å få denne så høy som mulig på begge modellene. Den siste modellen havnet på over 99% nøyaktighet noe som viser at det hjalp mye å legge til horisontalt vendte bilder og doble datasettet.

5.1.2 Trening av KNN og ECOC med tre klasser

Ved kjøring av applikasjonen med KNN eller ECOC trent på to klasser vil også output for hvert bilde være én av to klasser. Applikasjonen tar inn hele videoklipp og henter alle de rundt 700 bilderammene fra disse. En stor overvekt av disse, 90-95% er tomme, altså uten fisk. Det betyr at kun 5-10% av alle bilderammene inneholder den informasjonen applikasjonen skal klassifisere på bakgrunn av. De resterende må også klassifiseres, og havner dermed i én av klassene. Vi skjønte raskt at dette ville påvirke resultatet i negativ retning, så vi søkte etter en annen løsning. Etterhvert kom vi opp med to mulige løsninger; den ene var å automatisk filtrere ut de tomme bilderammene før klassifisering, slik at kun rammer med fisk stod igjen. Den første metoden kunne vært en enkel løsning om fiskene hadde vært i bildet til samme tid i hver video. Det er de ikke, og heller ikke like lenge om gangen. Dermed så vi for oss at å løse det problemet ville gi for mange problemer med kort tid igjen. Den andre var å klassifisere de tomme rammene som "Ingenfisk", altså introdusere en ny klasse. Med denne metoden er tanken å klassifisere alle rammene uten brukbar informasjon til en "søppelklasse", det betyr at vi ser bort fra alt som er klassifisert som "Ingenfisk". For å få til dette må man legge til en ny klasse i datasettet og trene hele nettverket på nytt. I samråd med veileder bestemte vi oss for å innføre en tredje klasse, "Ingenfisk".

Trening av KNN

Når vi startet ny trening med tre klasser valgte vi å fortsette med samme læringsalgoritme som ga oss best nøyaktighet med to klasser. Vi fikk også 10.000 bilder ekstra, som

gjør at selve treningen tar mye lengre tid. Vi endret noe på treningsinnstillinger da vi la til en ekstra klasse.

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',32, ...
    'MaxEpochs',15, ...
    'InitialLearnRate',1e-3, ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',5, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Listing 5.2: Treningsinnstillinger

På treningsinnstillingene 5.2, ser vi noe forskjell i forhold til innstillingene på 5.1. Vi har blant annet endret på "MiniBatchSize", "MaxEpochs" og "InitialLearnRate", men vi har også lagt til "LearnRateDropFactor" og "LearnRateDropPeriod" på den nye innstillingen vår.

LearnRateDropFactor

LearnRateDropFactor er en multiplikativ faktor som gjelder for læringsfrekvens hver gang et visst antall epoker går [28].

LearnRateDropPeriod

LearnRateDropPeriod multipliserer den globale læringsfrekvensen med en fallfaktor hver gang det angitte antall epoker går [28].

Nøyaktighet i %	KNN
1	99.15%
2	99.13%
3	98.84%
4	99.92%
5	98.78%
Gjennomsnitt	99.16%

Tabell 4: Fem treninger med KNN 3.klasse

På tabell 4, ser vi at vi har veldig god nøyaktighet over de fem treningene. Gjennomsnittet på 99.16% relativt høyt i forskjell fra treninger med to klasser. Det siste nettverket på 98.78% er det nettverket vi har benyttet på slutten mot klassifiseringer da det inneholder det nyeste datasettet.

Figur 24: 3.klasse KNN trening

På figur 24 ser vi bilde av den høyeste nøyaktigheten vi har fått ved trening av nettverk.

Trening av ECOC klassifiserer

Nøyaktighet i % KNN		ECOC
1	99.13%	99.74%
2	99.13%	99.87%
3	99.13%	99.87%
1	98.78%	99.82%
2	98.78%	99.84%
3	98.78%	99.87%

Tabell 5: 3.klasser ECOC

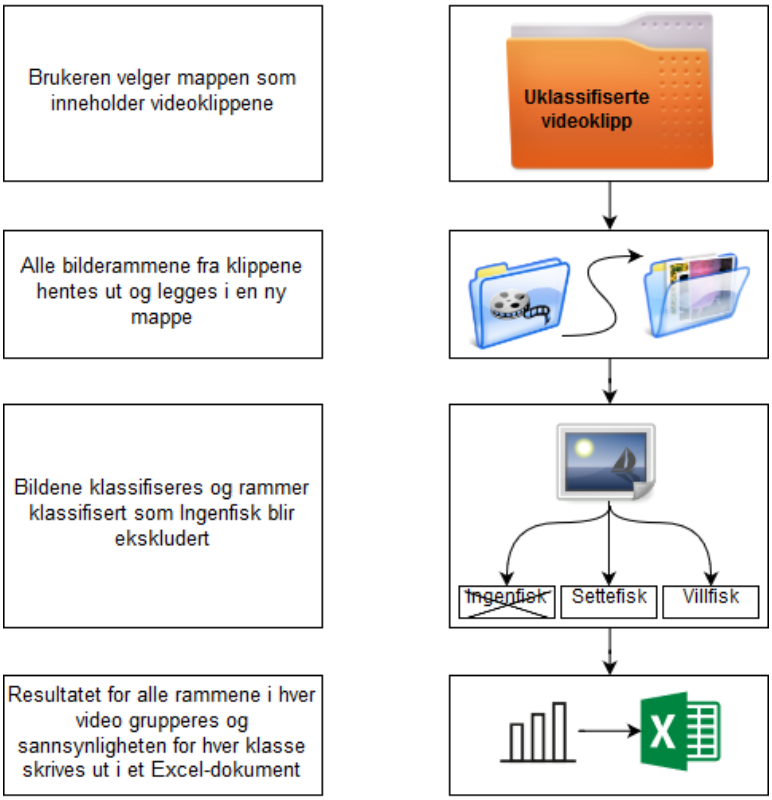
På tabell 5 ser vi hvilket resultat vi har fått med to forskjellige nettverk med nøyaktigheten 99.13% og 98.78%. Vi trente ECOC-klassifiseringen tre ganger for å få det beste resultatet, når vi kjører gjennom klassifiseringen får vi generelt høyere nøyaktighet enn det nettverket opprinnelig har.

Oppsummering av trening med tre klasser

Årsaken til at vi opprettet en ekstra klasse var på grunn av at applikasjonen tar inn hele videoklipp og henter alle de rundt 700 bilderammene fra hver enkelt klipp. Av 100% er det altså kun 5-10% vi er interesserte i, som er grunnlaget for klassifiseringen.
I treningen med tre klasser fortsatte vi med samme læringsalgoritme SGDM, men vi endret litt på treningsinnstillingene da vi fikk 50% større datasett. Vi tok også gjennomsnittet av fem treninger her også, som vi viste i tabell 4. Her hadde vi en gjennomsnittlig nøyaktighet på 99.16%, men vi hadde testet med litt forskjellig datasett der vi for eksem-

pel hadde lagt til bilder av halve fisker inn i klassen "Ingenfisk", slik at klassifiseringen skal bli så nøyaktig som mulig. Vårt beste resultat fra trening av nettverk har vi på figur 4.
Når vi hadde trent nettverket, gikk vi over på trening av ECOC-klassifisering hvor vi benyttet nettverkene 99.13% og 98.78% hvor vi oppnådde lik nøyaktighet på ECOC som var på 98.87% som vi kan se på tabell 5.

5.2 Utvikling av applikasjonen



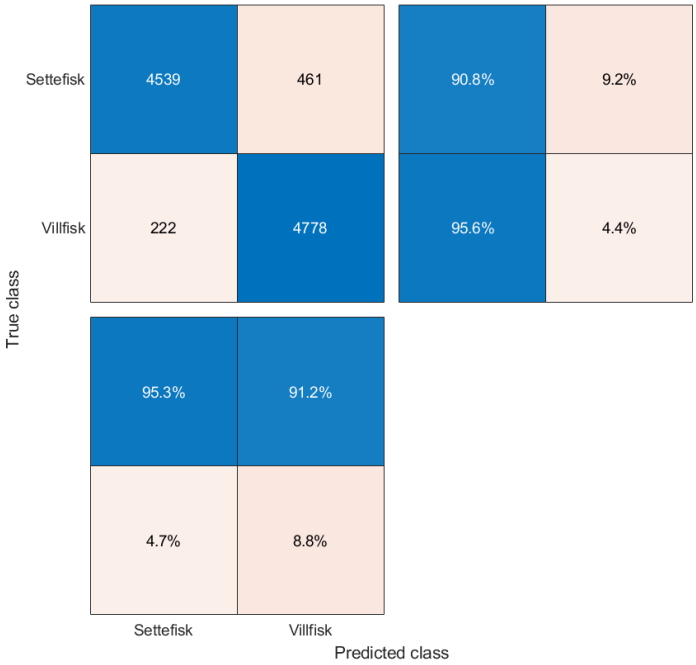
Figur 25: Flytskjema for applikasjonen

Applikasjonen består hovedsaklig av en trent KNN-modell, en ECOC-klassifiserer samt kode for å hente ut bilderammer fra videoene. Figur 25 viser applikasjonens flyt fra man begynner kjøringen til resultatet kommer ut til slutt.

5.2.1 Første utkast

Etter vi hadde trent nettverket var neste steg å teste ytelsen på ny data. Nøyaktigheten vi fikk ved trening var basert på en liten del, ca. 30% av datasettet vi trente på, som var satt av til validering. Den gir en pekepinn på hvordan ytelsen vil være for ny data, men gir ikke så mye informasjon om hvordan ytelsen er per klasse. Validering etter trening er det samme som å klassifisere ny data. Dette brukte vi dermed som utgangspunkt til å videreutvikle en applikasjon som ga oss mer informasjon. I første utkast kjørte vi på forhåndsklassifisert datasett, det samme som vi trente på. Det er slik det var mulig å få

nøyaktigheten til modellen, ved å sammenligne bildenes forhåndsbestemte klasse med resultatet fra klassifiseringen.



Figur 26: Feil-matrise med to klasser

Dermed kunne vi lage en confusion matrix eller feil-matrise som på figur 26. Denne sier noe om prestasjonen til modellen innen maskinlæring, altså hvor mange bilder som er klassifisert riktig og galt. Deretter kan man se på hver enkelt klasse for å si noe om ytelsen på disse:

• Sanne positive for settefisk Dette er bildene som er klassifisert som sanne i riktig klasse; settefisk klassifisert som settefisk. (Disse er sanne negative for villfisk)

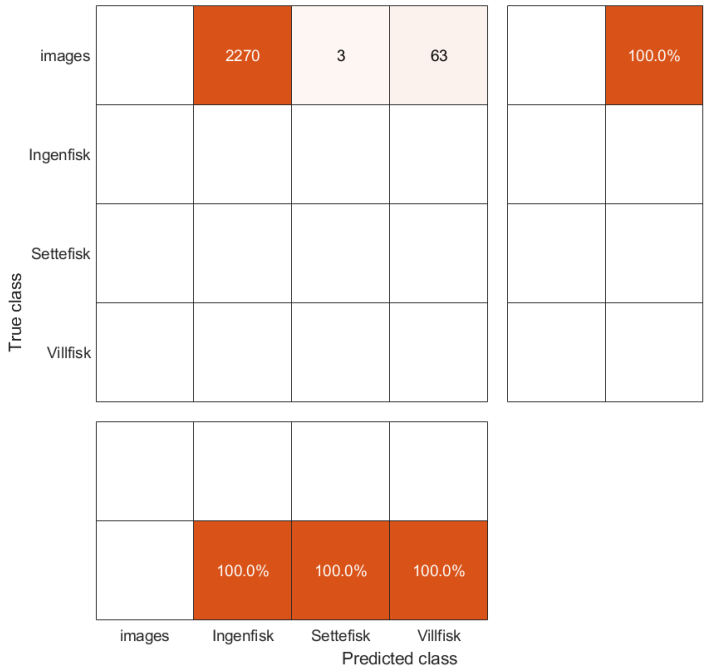
• Sanne negative for settefisk Dette er bildene som er klassifisert som usanne i riktig klasse; Ikke-settefisk klassifisert som i. (Disse er sanne positive for villfisk.)

• Falske positive for settefisk Dette er bildene som er klassifisert som sanne i feil klasse; villfisk klassifisert som settefisk. (Disse er falske negative for villfisk)

• Falske negative Dette er bildene som er klassifisert som usanne i feil klasse; settefisk klassifisert som villfisk. (Disse er falske positive for villfisk.)

5.2.2 Andre utkast

Vi hadde til dette tidspunktet kun klassifisert forhåndssortert data. Det ville ikke fungert i praksis, da dette må gjøres manuelt på forhånd. For å løse dette problemet måtte vi gjøre om applikasjonen slik at vi kunne mate inn hele videoklipp. Inntil dette tidspunktet hadde vi hentet ut alle bilderammene via et script i Python, deretter har vi lagt dem i sine tilhørende mapper som utgjorde datasettet. Det viste seg å være enkelt å implementere dette direkte i Matlab, og deretter bruke den resulterende mappen til klassifisering. Dermed fikk vi et nytt problem. Alle bildene lå nå usortert i samme mappe.



Figur 27: Feil-matrise med tre klasser, usortert

Feil-matrisa ga oss ikke lenger noe særlig informasjon. Den viste fortsatt hvor mange som ble klassifisert til hver klasse, men ikke lenger hvilken klasse de egentlig tilhører som man kan se på figur 27. Ved kjøring av funksjonen som gjør klassifiseringen i Matlab legges resultatet for hvert bilde i en listearray. Dette resultatet er en av de tre klassenes navn. Det sa oss ikke mye uten at vi visste hvilket bilde det var snakk om. Vi hentet så ut filnavnet til alle bildene som ble laget fra alle filmene og la dette i en ny array. Filnavn og resultatet ble satt sammen til én tabell. Dermed hadde vi alt av resultater i én liten tabell, og på denne måten kunne vi se nøyaktig hvilke bilderammer som ble feilklassifisert.

5.2.3 Automatisk gjenkjenning

Andreutkastet fungerte fint når det kom til å forbedre datasettet og teste forskjellige treningsinnstillinger, men vi kunne ikke levere en applikasjon hvor brukeren måtte telle opp antall rammer i hver klasse og regne ut sannsynligheten manuelt. Vi måtte tilbake i tankeboksen. For brukeren er det interessante hva totalen er for hele videoklippet, ikke resultatet for hver enkelt ramme. For å få ut totalresultatet hadde vi til nå måttet legge tabellen inn i f.eks. Excel og gruppert etter videonummer før vi talte opp.

Det vi hadde å gå på var filnavn og hver rammes resultat. Filnavnet var det vi som ga til hvert bilde, så vi hadde allerede laget dette formatet: "Videonummer_Rammenummer.png", f.eks. "22_440.png". For å gruppere resultatet for hver video måtte vi skille ut videonummeret i hver fil. Dette gjorde vi ved å søke etter første instans av "_" i filnavnet og trekke ut det forestående. Da kunne vi iterere over alle filene hvor videonummeret var det samme. Det som gjenstod da var å telle opp alle forekomster av "Villfisk" og "Settefisk" og regne ut sannsynligheten for hver klasse. Her fikk vi god bruk for det vi hadde lært i programmeringsemnene tidligere i utdanningen. Et utdrag fra koden vises under i kodeblokken 5.3.

```
firstFile = fname(1,:);
currentVid = firstFile(1:strfind(fname(1,:), '_')-1);
resultater = cell(numVideos,5);
indx = 1;
for j = 1 : numFiles
    nextFile = fname(j,:);
    nextVid = nextFile(1:strfind(fname(j,:), '_')-1);
    percentSettefisk = 0;
    percentVillfisk = 0;
    if (~strcmp(currentVid, nextVid) || j == 1)
        currentVid = nextVid;
        antVillfisk = 0;
        antSettefisk = 0;
        for m = 1 : numFiles
            vidFrame = fname(m,:);
            vidFrameVid = vidFrame(1:strfind(fname(m,:), '_')-1);
            if strcmp(vidFrameVid, currentVid)
                if strcmp(char(lables(m,1)), "Villfisk")
                    antVillfisk = antVillfisk + 1;
                elseif strcmp(char(lables(m,1)), "Settefisk")
                    antSettefisk = antSettefisk + 1;
                end
            end
        end
        percentSettefisk = antSettefisk/(antSettefisk + antVillfisk)*100;
        percentVillfisk = antVillfisk/(antSettefisk + antVillfisk)*100;
        if antSettefisk == 0 && antVillfisk == 0
            percentSettefisk = 50;
            percentVillfisk = 50;
        end
        if percentSettefisk < 80 && percentVillfisk < 80
            usikker = "Denne er usikker!";
        else
            usikker = "";
        end
        resultater(indx, :) = {currentVid, antVillfisk, percentVillfisk, antSettefisk, percentSettefisk, usikker};
        indx = indx + 1;
    end
end
```

Listing 5.3: Koden for sammensetning av resultatene i applikasjonen

Vi benytter nestede løkker og går gjennom hver fil og sjekker videonummeret. Deretter sjekker vi alle filene om de har samme videonummer. Hvis filen har samme videonummer sjekkes det om den er klassifisert som villfisk eller settefisk, mens ingenfisk ikke telles med. Når alle filenes resultater under samme video er talt opp regnes sannsynligheten for hver av klassene ut. Om begge klassene har 0 klassifiserte settes begges

sannsynlighet til 50%. Hvis sannsynligheten for begge klassene er under 80% settes videoen som "usikker" som betyr at denne bør gåes over manuelt for å se om klassifiseringen stemmer. Til slutt settes resultatene sammen i en tabell med videonummer, antall av hver klasse, sannsynligheten for hver klasse og om den er "usikker". Det er denne tabellen som skrives ut i et Excel-ark ved endt kjøring av applikasjonen.

5.3 Maskinvare

Vi har primært benyttet oss av to datamaskiner ved kjøring av nettverk.
Den ene er en stasjonær datamaskin som et gruppemedlem eier og den andre datamaskinen er en maskin vi har fått tilgang på via NTNU i Gjøvik. Vi har samkjørt disse datamaskinene slik at de benytter samme treningsinnstillinger som vi ser nede på listen 5.4.

Maskin 1 - HjemmePC
Denne bruker GPU ved kjøring av Matlab applikasjoner.

- GPU
  GeForce GTX 1080Ti

Maskin 2 - PC på NTNU i Gjøvik
Denne bruker CPU ved kjøring av Matlab applikasjoner.

- CPU
  Intel i7-8086K @ 4.00GHz

```
options = trainingOptions('sgdm', ...
    'MiniBatchSize',32, ...
    'MaxEpochs',15, ...
    'InitialLearnRate',1e-3, ...
    'LearnRateDropFactor',0.1, ...
    'LearnRateDropPeriod',5, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',3, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Listing 5.4: Treningsinnstillinger

Results	
Validation accuracy:	98.09%
Training finished:	Reached final iteration
Training Time	
Start time:	13-May-2019 22:45:17
Elapsed time:	942 min 10 sec
Training Cycle	
Epoch:	15 of 15
Iteration:	9840 of 9840
Iterations per epoch:	656
Maximum iterations:	9840
Validation	
Frequency:	3 iterations
Patience:	Inf
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.001

(a) GPU

Results	
Validation accuracy:	98.78%
Training finished:	Reached final iteration
Training Time	
Start time:	06-May-2019 14:53:28
Elapsed time:	4408 min 33 sec
Training Cycle	
Epoch:	15 of 15
Iteration:	9840 of 9840
Iterations per epoch:	656
Maximum iterations:	9840
Validation	
Frequency:	3 iterations
Patience:	Inf
Other Information	
Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.001

(b) CPU

Figur 28: GPU vs CPU

Ved første øyekast ser vi at på figur 28 har tilnærmet samme nøyaktighet. Den store forskjellen er nok "Elapsed time", altså tidsbruken det tar å kjøre gjennom nettverket. På figur 28a, ser vi at den har benyttet 942 minutter i motsetning til figur 28b som har 4408 minutter. Forskjellen blir 3466 minutter, som tilsvarer nesten 58 timer, selv om begge maskinene er kraftig utstyrt er det likevel en stor forskjell.
Et lite utdrag fra en artikkel vi fant når vi undersøkte komponenter i forbindelse med maskinlæring bekrefter det vi har sett:
"The GPU is just the heart of deep learning applications - the improvement in processing speed is just too huge to ignore."[29]

5.4 Oppsummering og diskusjon av implementering og realisering

I dette kapittelet startet vi med trening av nettverk i Matlab. Vi benyttet AlexNet som er et ferdigtrent nettverk, hvor vi i første omgang trente med to klasser. Etter mange treninger med to klasser og forskjellige læringsalgoritmer fant vi en innstilling og en algoritme som ga oss tilfredsstillende nøyaktighet. Oversikten av treningene med to klasser kan du se på tabell 1 og 2, hvor vi fikk et gjennomsnitt på 94.86% ved bruk av SGDM uten CLAHE. Når vi hadde trent ferdig nettverk, trente vi en ECOC-klassifiserer som reduserer et klassifiseringsproblem med n-klasser til flere små binære problemer. Vi økte nøyaktigheten betraktelig fra de to nettverkene vi opprinnelig testet fra som man ser på tabell 3.

Vi fant ut underveis i implementeringen at når vi kjører KNN eller ECOC trent på to klasser vil output for hvert enkelt bilde være én av to klasser, da applikasjonen henter rundt 700 bilderammer fra et enkelt videoklipp, betyr det at det er kun 5-10% av bilderammene som inneholder den informasjonen vi er interesserte i, det vil si at de tomme bilderammene vil bli klassifisert som enten settefisk eller villfisk, som kan gi oss feil resultat. Vi opprettet en ny klasse "Ingenfisk", hvor vi samlet bilderammene som ikke inneholdt fisk, slik at klassifiserings-applikasjonen vår kan klassifisere de tomme bilderammene i en egen klasse. Siden vi la til en ekstra klasse, måtte vi kjøre treningen av nettverket på nytt med tre klasser.

Når vi startet ny trening av nettverket med tre klasser fortsatte vi med samme læringsalgoritme (SGDM), siden vi fikk 50% større datasett måtte vi gjøre noen små endringer på treningsinstillingene. De oppdaterte treningsinnstilligene kan man se på listen 5.2, hvor det er noe forskjell fra innstillingene vi brukte med to klasser i starten som man ser på listen 5.1. Vi kjørte også treningen fem ganger på de nye innstillingene og tok gjennomsnittet. Vi fikk betraktelig mye bedre gjennomsnitt enn vi hadde fått tidligere, gjennomsnittet ble 99,16% som man kan se på tabell 4. Testingen med ECOC-klassifiserer ble gjennomført med det siste nettverket vårt på 98.78%, der vi fikk en klassifiseringsnøyaktighet på 99.87% som vi er fornøyde med.

Utviklingen av applikasjonen skjedde gradvis, og i takt med treningen. Vi brukte applikasjonens resultater til å forbedre treningene som igjen forbedret applikasjonen. Det største problemet var å få gruppert resultatene for den enkelte video, men det lot seg gjøre med litt teknikker fra de tidligere programmeringsemnene. Dette var forsåvidt den eneste kodingen i prosjektet som var utfordrende, da det meste var kun bruk av enkle, ferdiglagde funksjoner. Vi er meget fornøyd med hvordan applikasjonen endte opp, og mener den gir et oversiktelig bilde over resultatene i et velkjent format som Excel-dokumenter er. I tillegg har man bilderammene lagret etter kjøring for de som er merket "usikker" slik at man enkelt kan se over resultatet.

6 Resultater

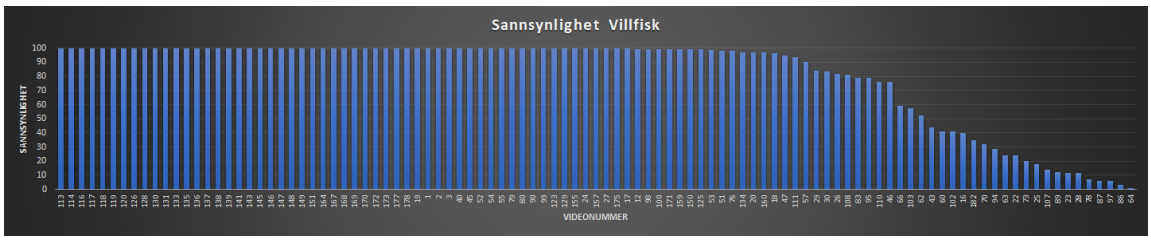
I dette kapittelet vil man få et innblikk i hvilke resultater vi har fra klassifiseringen med KNN og ECOC. Vi tar også med noen bilder av fisk som blir klassifisert feil. Resultatene oppsummeres underveis, siden det blir mange figurer med forskjellige resultat.

6.1 Maskinlæring - testing

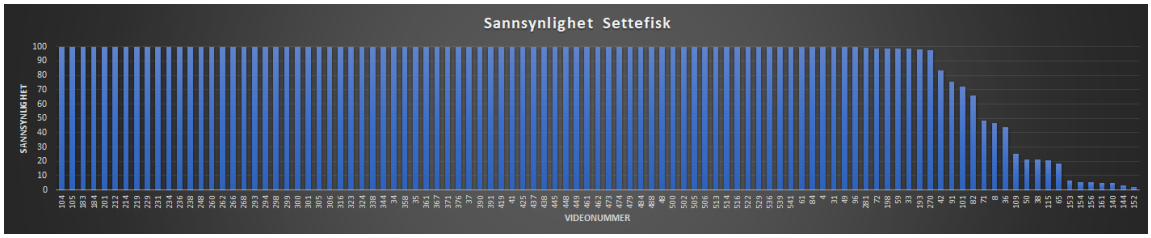
I denne seksjonen skal vi ta for oss testingen av ECOC-klassifiserere og KNN. De to modellene har hver sin applikasjon som virker på samme måte. Alle tester er gjort på samme datasett, dvs. alle videoene vi fikk tildelt av NINA ved prosjektstart. Resultat-histogrammene er laget utfra resultatet som kommer etter kjøring av applikasjonene i Excel. Vi foreslår at de som bli klassifisert under 80% må behandles manuelt, slik at NINA må vurdere om det er villfisk eller settefisk. Om det blir klassifisert feil med over 80% setter vi den som feilklassifisert, og er følgelig det som er viktigst å unngå. Tiden det tar å klassifisere fra man trykker start til man har fått et ferdig resultat er rundt 25 minutter.

6.1.1 Testing av KNN og ECOC med to klasser

Testing av KNN med to klasser



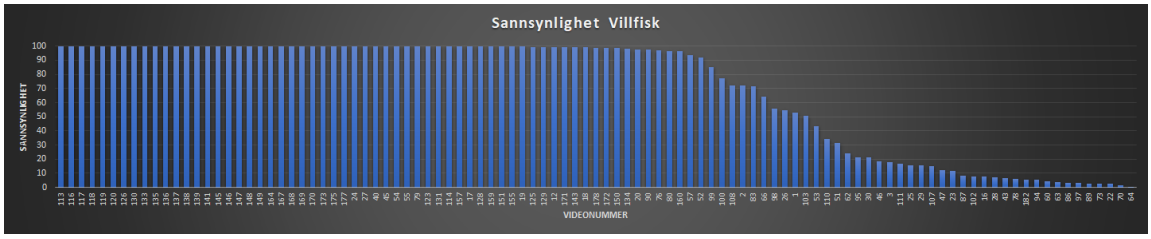
(a) Sannsynligheten for villfisk, kjøretid 1550 sekunder



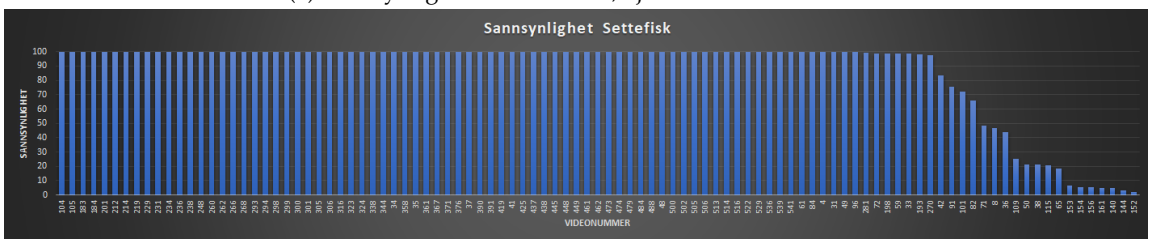
(b) Sannsynligheten for settefisk, kjøretid 1393 sekunder

Figur 29: Histogram av KNN, to klasser

Testing av ECOC classifier med to klasser



(a) Sannsynligheten for villfisk, kjøretid 1403 sekunder



(b) Sannsynligheten for settefisk, kjøretid 1400 sekunder

Figur 30: Histogram av ECOC classifier, to klasser

Oppsummering

Figur 29:

- 29a
  76 som er klassifisert riktig
  17 som er under 80% nøyaktighet
  10 som har blitt klassifisert feil

- 29b
  79 som er klassifisert riktig
  10 som er under 80% nøyaktighet
  12 som har blitt klassifisert feil

Figur 30:

- 30a
  63 som er klassifisert riktig
  15 som er under 80% nøyaktighet
  25 som har blitt klassifisert feil

- 30b
  83 som er klassifisert riktig
  10 som er under 80% nøyaktighet
  8 som har blitt klassifisert feil

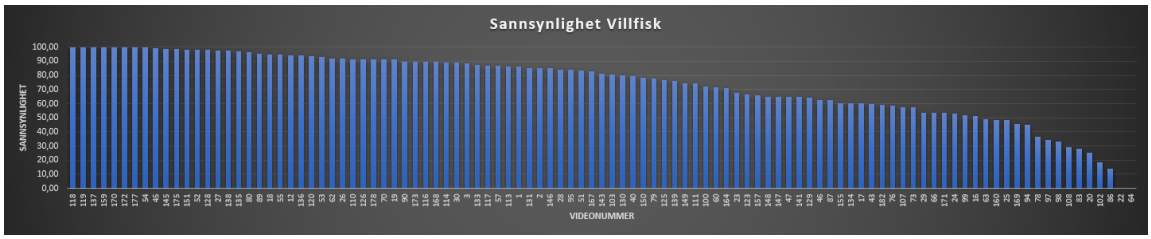
På figur 29 og 30 har vi testet KNN og ECOC-klassifiseringer med to klasser. Vi kjørte klassifiseringer på to klasser på nytt mot slutten av innlevering for å presentere resultatet vårt på en pen måte, slik at vi kan se forskjellen mellom to og tre klasser da applikasjonen ikke var ferdig på dette tidspunktet. Nøyaktigheten på ECOC-klassifisereren var på 99.90%, mens nettverket hadde en nøyaktighet på 99.15%. Vi ser altså at ECOC-klassifisereren plukker opp bedre egenskaper enn nettverket. Dette kan være fordi den videretrenes på egenskapene fra nettverket på det samme datasettet, og dermed ser mønstre som nettverket ikke rakk å lære under treningen. Det er mulig at man ved å optimalisere treningsinnstillingene til nettverket kan oppnå samme resultat der.

6.1.2 Testing av KNN og ECOC med tre klasser

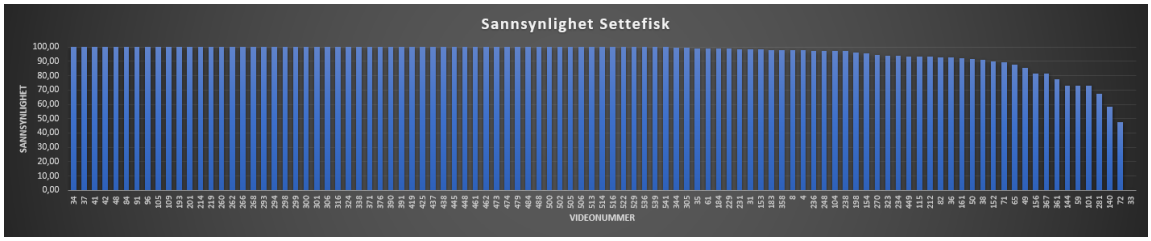
I denne seksjonen vil vi se nærmere på resultatene våre gjort med de tre klassene settefisk, villfisk og ingenfisk.

Testing av ECOC klassifiserer med tre klasser

Første resultat med tre klasser



(a) Sannsynligheten for villfisk



(b) Sannsynligheten for settefisk

Figur 31: Histogram av ECOC classifier, tre klasser, første datasett

Oppsummering

- 31a
  54 som er klassifisert riktig
  45 som er under 80% nøyaktighet
  4 som har blitt klassifisert feil

- 31b
  93 som er klassifisert riktig
  7 som er under 80% nøyaktighet
  1 som har blitt klassifisert feil

Første resultat med tre klasser på alle videoene var som man kan se på figur 31 veldig ujevnt for de to klassene. Ytelsen på settefisk var god, mens villfisk ga nærmere 50% feil- og uklassifiserte videoer. Dette tydet på at modellen hadde lært gode egenskaper ved settefisk, men ikke ved villfisk. Vi gjennomgikk bilderammene som var klassifisert feil ved mange av villfisk-videoene, bl.a. video 22 og 64 som ikke hadde noen rammer i riktig klasse, og det var to ting som utpekte seg:

(a) Feilet villfisk 1

(b) Feilet villfisk 2

(c) Feilet ingenfisk 1

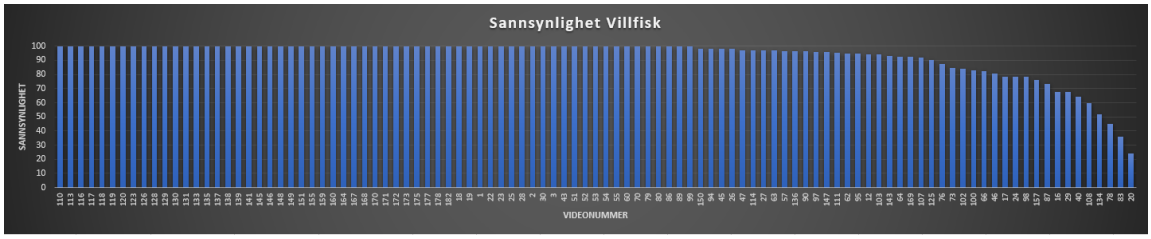
(d) Feilet ingenfisk 2

Figur 32: Eksempler på feilklassifiserte bilderammer

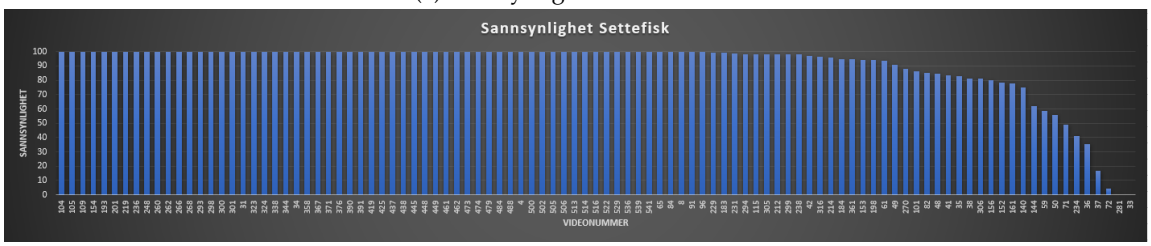
Det vi fant, som vist på figur 32, var at videoene av villfisk med små fisker som figur 32a fra video 64 og 32b fra video 22 ofte ble klassifisert feil. Dette tenkte vi var på grunn av filtreringen vi gjorde av datasettet i starten av prosjektet. Vi hadde valgt de beste bilderammene, men i noen av klippene var det vanskelig å velge noen i det hele tatt. Dermed var ikke alle klippene representert i datasettet, og spesielt de med uklare bilder av de små fiskene ble filtrert bort. Det andre vi fant, som vist på figur 32d og 32c, var at bilderammer hvor kun halve fisken var synlig uten at fettfinnen var i bildet oftest ble klassifisert som settefisk. Dette var et problem vi ikke hadde tenkt på før vi testet mot hele videoklipp, og innførte den tredje klassen, ingenfisk. Da vi la til denne klassen hentet vi som sagt i kapittel 4 de 25 første rammene i hver video. Disse var helt tomme, så rammer hvor noe av fisken var synlig havnet i en av de andre klassene. Men siden mange av disse altså ikke viste fettfinnen ga dette et feilaktig resultat. Det at de fleste av disse ble klassifisert som settefisk tror vi var tilfeldig.

Det vi endte opp med å gjøre for å utbedre resultatet var dermed å: **1.** Legge til nye bilderammer i villfisk-klassen i datasettet. Dette var hovedsaklig rammer fra videoklipp med små fisker som vi hadde filtrert bort ved første filtrering av datasettet. **2.** Legge til nye bilderammer i ingenfisk-klassen i datasettet. Dette var rammer hvor kun deler av fiskene var synlig slik at riktig klassifisering ikke kunne identifiseres.

Andre resultat med tre klasser - Oppdatert datasett



(a) Sannsynligheten for villfisk



(b) Sannsynligheten for settefisk

Figur 33: Histogram av ECOC classifier, tre klasser, andre datasett

Oppsummering

- 33a
  90 som er klassifisert riktig
  13 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

- 33b
  90 som er klassifisert riktig
  9 som er under 80% nøyaktighet
  3 som har blitt klassifisert feil

Resultatet vi fikk etter å ha oppdatert datasettet nevnt i forrige seksjon kan sees på figur 33. Der ser vi en klar forbedring på villfisk som gikk fra ca 50% riktig til nærmere 90%. Derimot fikk vi en liten forverring på settefisk. Dermed undersøkte vi de rammene fra de klippene som gjorde det dårligst fra settefisk. Igjen var det noe som utpekte seg:

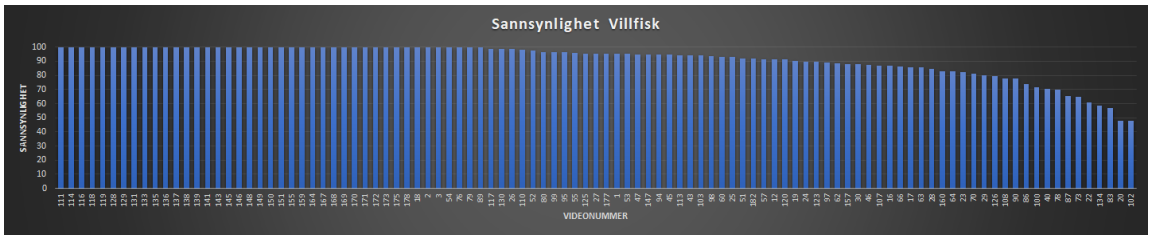
(a) Feilet settefisk 1

(b) Feilet settefisk 2

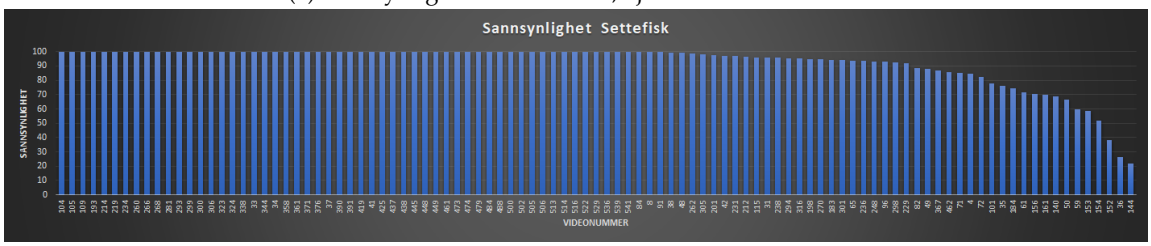
Figur 34: Eksempel feilet settefisk

Her fant vi noe lignende som etter første resultat. Figur 34 viser én ramme fra de to videoene som ikke hadde noen rammer i riktig klasse, video 33 og 281. Igjen så vi at modellen slet med de små fiskene, men denne gang i settefisk-klassen. Dette kunne tyde på at modellen hadde plukket opp at det var en overvekt av små fisker i villfisk, og dermed klassifiserer disse som villfisk. Vi gjorde som sist og la til nye bilderammer, men denne gang i settefisk-klassen. Også denne gang av de små fiskene som feilet.

Siste resultat med tre klasser - Siste oppdatering av datasett



(a) Sannsynligheten for villfisk, kjøretid 1667 sekunder



(b) Sannsynligheten for settefisk, kjøretid 1484 sekunder

Figur 35: Histogram av ECOC, tre klasser, siste datasett

Oppsummering

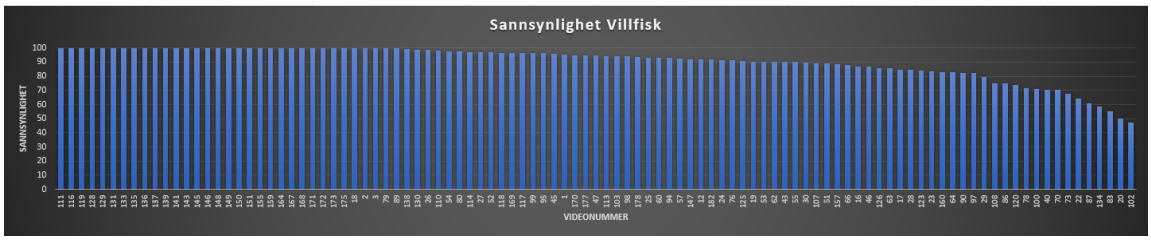
Figur 35,

- 35a
  90 som er klassifisert riktig
  13 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

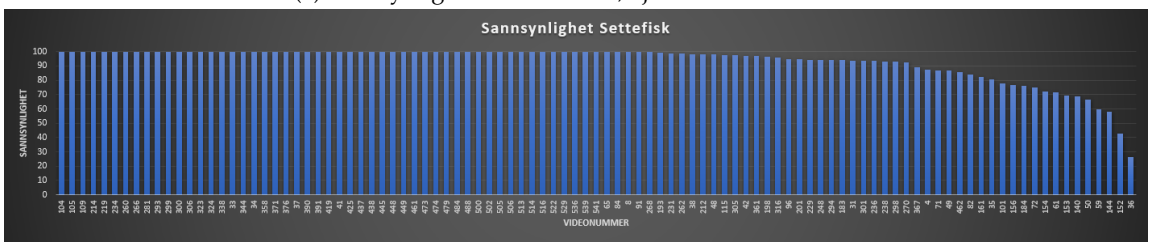
- 35b
  90 som er klassifisert riktig
  14 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

Her har vi brukt en ECOC-klassifiserer med en nøyaktighet på 99.87% trent fra KNN med nøyaktighet 98.78%. Hvis man ser på figur 35, så ser man at resultatet for begge klasser er mye jevnere enn tidligere. Nøyaktigheten for begge klasser er rett under 90% og vi fikk fjernet alle feilklassifiseringer. Her så vi i praksis hvor viktig det er å ha et balansert datasett ved dyp læring. Modellen ser mønstre i dataene, og ser raskt om det er noe som stikker seg ut, som en overvekt av små fisk i en av klassene.

Testing av ECOC med tre klasser - Klassifisering av JPEG



(a) Sannsynligheten for villfisk, kjøretid 673 sekunder



(b) Sannsynligheten for settefisk

Figur 36: Histogram av ECOC, tre klasser - JPEG, kjøretid 684 sekunder

Oppsummering

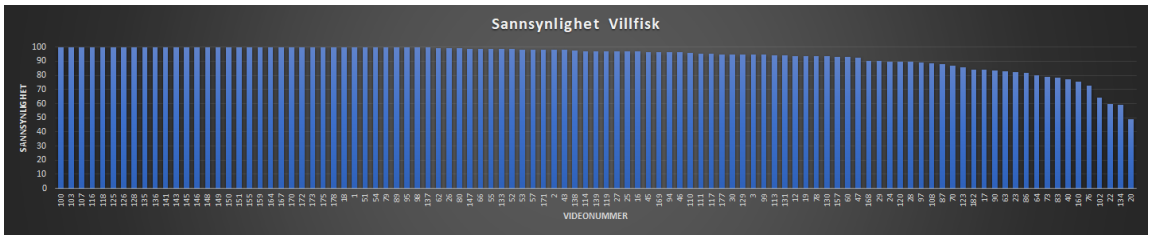
Figur 36,

- 36a
  88 som er klassifisert riktig
  15 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

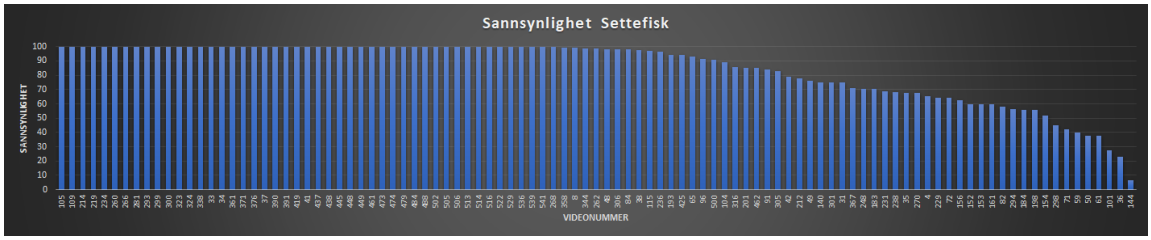
- 36b
  88 som er klassifisert riktig
  13 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

På figur 36 ser vi resultatet av den samme modellen, men hvor applikasjonen er satt til å lagre alle bilderammene i JPEG-format. Dette fører til komprimering av bildet, og vi ser at det har en liten, negativ innvirkning på resultatet. Modellen er ikke trent på JPEG-bilder. Her er det verdt å merke seg kjøretiden som er betraktelig lavere enn ved lagring i PNG-format. Dette er fordi bildene blir mindre i JPEG-format og krever mindre skriving til disk, som er tidkrevende.

Testing av KNN med tre klasser - Siste oppdatering av datasett



(a) Sannsynligheten for villfisk, kjøretid 1442 sekunder



(b) Sannsynligheten for settefisk, kjøretid 1421 sekunder

Figur 37: Histogram av KNN

Oppsummering

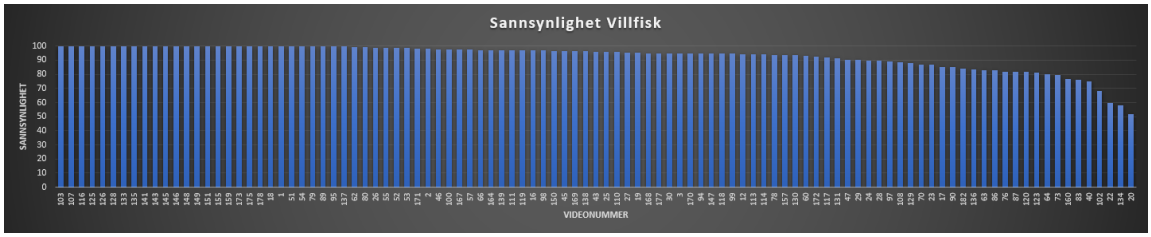
Figur 37:

- 37a
  94 som er klassifisert riktig
  9 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

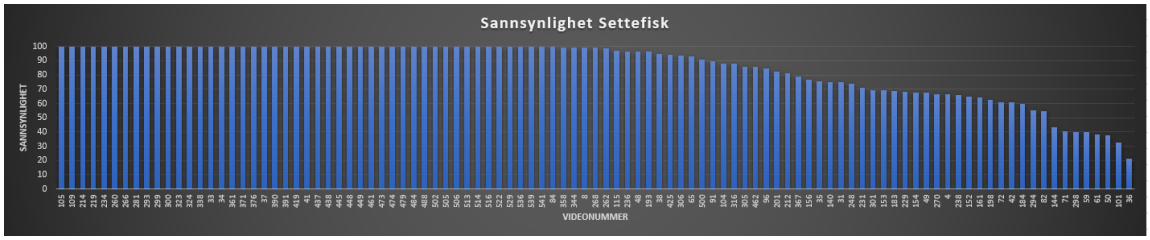
- 37b
  67 som er klassifisert riktig
  33 som er under 80% nøyaktighet
  1 som har blitt klassifisert feil

Her har vi testet KNN-klassifisering, nøyaktigheten på nettverket var på 98.78%. På figur 37 ser vi at nettverket er litt bedre på villfisk enn ECOC på figur 35. Nettverket er derimot veldig mye svakere på settefisk. Ettersom vi hadde basert applikasjonen på ECOC-klassifiserere i store deler av prosjektet har vi ikke sett mye på resultatet av KNN annet enn ved treningen. Det viser seg at ECOC kan plukke opp egenskaper ved bildene som ikke nettverket har plukket opp. Dette kan igjen være mulig å forbedre ved å endre treningsinnstillinger ved trening av nettverket.

Testing av KNN med tre klasser - Klassifisering av JPEG



(a) Sannsynligheten for villfisk



(b) Sannsynligheten for settefisk

Figur 38: Histogram av KNN - JPEG

Oppsummering

Figur 38,

- 38a
  95 som er klassifisert riktig
  8 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

- 38b
  69 som er klassifisert riktig
  32 som er under 80% nøyaktighet
  0 som har blitt klassifisert feil

På figur 38 ser vi kjøring med nettverket mot JPEG-bilder. Her ser vi, i motsetning til ved ECOC, en liten forbedring i resultatet. Dette kan tyde på at nettverket er mer generalisert med tanke på dataene.

6.1.3 Diskusjon av resultater

Det siste nettverket med tre klasser og siste oppdatering av datasett er fra 6. mai, og egentlig langt ut i rapportskrivingen. Dette gjorde at skrivingen ble litt forskjøvet så vi måtte bare gi oss med optimalisering av nettverket og ECOC. Men vi er veldig fornøyd med resultatene selv om man alltid kan gjøre små forbedringer. Vi har sett en jevn forbedring i resultatene fra start til slutt, og har lært mye om hvordan KNN og ECOC virker.

På figur 38 og 36 så vi at det var litt forskjell ved klassifisering av JPEG- og PNG-bilder, men dette gikk begge veier så det var ikke lett å trekke noen generell konklusjon. Dette viste seg dermed å stemme greit overens med det vi hadde sett av tidligere arbeid, at så lenge komprimeringen holdes på et lavt nivå vil det ha liten innvirkning på resultatet [30].

7 Utførelse

Dette kapittelet inneholder sammendrag fra hver enkelt periode i prosjektet, kapittelet beskriver prosessen av arbeidet vårt.

7.1 Oppstart

7.1.1 Prosjektplanlegging

Periode 7.januar - 2.februar
Bacheloroppgaven startet med å prosjektere og skrive prosjektplanen for prosjektet. Etter første møte med veileder fikk vi grunnleggende informasjon rundt bachelorarbeid og hvordan vi skal starte med prosjektplanen. Vi kontaktet oppdragsgiver Norsk institutt for naturforskning (NINA) tidlig i oppstartsfase, men siden det var sykdom i starten fikk vi ikke hatt møte med oppdragsgiver og signert prosjektavtale før den 29.01.2019. I møtet med oppdragsgiver fikk vi litt mer forståelse av hva de ønsket og krevde av oss.

7.1.2 Egenlæring

Periode:4.februar - 18.februar
Hovedfokuset i starten er å tilegne nytt lærestoff. Vi satte av to uker til egenlæring da det var nytt læringsstoff å sette seg inn i. Vi ble i møte med veileder oppfordret til å se på ulike teknikker for datasyn, samt lese tidligere litteratur hvor lignende arbeid har blitt gjennomført fra tidligere.

Verktøy
Læringsverktøy for prosjektet sto mellom OpenCV og Matlab, etter første møtet med veileder ble vi henvist mer mot Matlab som læringsverktøy da NTNU har lisens for programvaren og at veileder har god kunnskap innen programvaren, samtidig som at det finnes mye eksempler og læringsstoff for programmet.
Vi testet forskjellige kodesnutter som lå tilgjengelig på internett for å forstå hvordan det fungerer og lærte hvordan Matlab fungerte. Resten av egenlæringsperioden ble brukt til å se på litteratur og forskjellige teknikker brukt innen Matlab.

7.2 Scrum-prosessen

Periode:18.02.2019 - 29.04.2019
Her vil vi gå igjennom hva som har blitt gjort i hver sprint.

7.2.1 Sprint 1

Periode:18.02 - 04.03
Hovedfokuset til den første sprinten var å få tilgang på datasettet og bearbeide det.
Datasett
Det tok litt tid før vi mottok datasettet fra NINA grunnet sykdom. Når vi mottok dette

fikk vi totalt 204 videosnutter. Vi laget et Python script som hentet ut alle bilderammene av videosnuttene, forsøkte videre å automatisere prosessen med å filtrere ut bilderammene vi ikke har bruk for, altså de rammene som ikke inneholder bilde av fisk. Det var ikke så lett som vi trodde, endte dermed opp med å gjøre dette manuelt. Til slutt satt vi igjen med 10.000 med settefisk/villfisk, 5.000 fordelt på hver. I denne perioden hadde vi ingen veiledningsmøte da vi arbeidet med datasettet vårt og hadde ingen generelle spørsmål rundt dette.

7.2.2 Sprint 2

Periode: 4.mars - 18.mars
Hovedfokuset denne perioden var å gjøre datasettet større ved å tilføye ulike teknikker.

Bildebehandling
Vi startet med veiledningsmøte den 5.mars, der vi gjennomgikk forskjellige teknikker for datasettet vårt. Veileder ba oss teste forskjellige fargekanaler og legge til forskjellige kontraster slik at det blir så høy kontrast som mulig mellom fisken og bakgrunnen.
Veileder kom også med et tips vedrørende filformatet på bildene vi konverterte fra videosnuttene, vi opprettet opprinnelig med JPEG/JPG, men siden JPG er komprimerte bilder av videosnutten, kan vi miste viktig detaljer i selve bilde når vi gjør maskinlæring. Vi opprettet dermed nytt datasett med PNG som filtype den ikke komprimerer, den er original mot videosnutten.

Augmenteringsteknikker
Etter vi hadde filtrert på nytt igjen med datasettet så vi på forskjellige teknikker for augmentering.

- Contrast-limited adaptive histogram equalization (CLAHE)
- Binært bilde
- Gråskala
- Fargekanaler
- Uskarphet
- Beskjæring og skalering
- Rotering av bilder

Vi startet også med rapportskriving, da vi var i lynkurs med foreleser Frode Haug om hvordan vi skulle skrive og fremstille rapporten. Etter lynkurset startet vi på innledning, da det var mye vi kunne legge inn fra prosjektplanen vi hadde laget. Vi ble også bedre kjent i LaTeX med forskjellige skrivemåter og hvordan rapporten skal struktureres.

7.2.3 Sprint 3

Periode:18.mars - 1.april
I løpet av denne sprinten skulle vi ha en "fungerende prototype" til den 1. april.
Hovedfokuset for denne sprinten var å teste forskjellige ferdigtrente nettverk og lage et script som tar i mot datasettet vårt og klassifiserer det.

Bilde klassifisering
Vi startet med å se på ferdigtrente nettverk, henholdsvis AlexNet og GoogLeNet. Vi leste oss opp på nettverkene og testet med datasettet vi hadde. Etter noen dager med

testing fant vi ut at vi skulle gå for AlexNet. Største grunnlaget for valget vårt falt på tid og nøyaktighet. Der vi trengte et nettverk som var nøyaktig og ikke brukte så lang tid på trening av nettverket. AlexNet fikk bedre nøyaktighet og brukte mindre tid enn GoogLeNet.
Jobbet videre med å utvikle en applikasjon hvor vi brukte AlexNet som ferdigtrent nettverk. Når vi hadde utviklet applikasjonen kjørte vi programmet slik at vi fikk nøyaktighet av nettverket brukt med vårt datasett. Den 20. mars fikk vi i gang en "fungerende prototype". Videre så vi på ligende arbeid som har blitt gjort før og hvilken nøyaktighet de fikk ved klassifisering, slik at at vi fikk en pekepinn på hvordan vi lå med nøyaktigheten vi hadde fått.

7.2.4 Sprint 4

Periode:1.april - 15.april
Hovedfokuset denne sprinten var å validere nettverket og se om vi fikk gjenkjent bilder som feiler i klassifiseringen.

Resultater av teknikker
Vi hadde ordnet en tabell med oversikt hvor vi hadde kjørt tre ulike teknikker fem ganger og sett på hvilken teknikk som var den beste og hadde best nøyaktighet. Teknikkene ble kjørt fra privat PC og via fargelab på NTNU i Gjøvik, der vi fikk lånt en stasjonær datamaskin. Vi spurte også NINA om de hadde noen innsikt i hvilket kamera som er plassert i fisketrappen, da det ikke er betydlig forskjell mellom fargekanalene RGB, som kan tyde på et infrarødt kamera, men dette har vi ikke fått bekreftet.

Gjenkjenne bilder som feiler
For å kunne gjøre nøyaktigheten vår bedre måtte vi se hvilken videosnutt av fisk som ble klassifisert feil. Da opprettet vi en tabell med oversikt over fiskene som ble klassifisert som feil og forsøkte å se om vi fant en årsak til dette. Blant de som ble klassifisert som feil var det oftest en liten fisk som man omtrent ikke så, eller at det var for mye forstyrrelser (luftbobler) i bildet.

7.2.5 Sprint 5

Periode:15.april - 29.april
Fokuset på denne sprinten var å opprette en ekstra klasse i nettverket, klassifisereren og datasettet vårt.

Tre klasser for klassifisering
Vi opprettet en ekstra klasse, "Ingenfisk" hvor vi la inn 10.000 bilder av bilderammer hvor det ikke er fisk. Resultatet av nettverket etter vi kjørte med tre klasser ble over 99%. Laget videre en ECOC-klassifiserer fra nettverket, som også endte på over 99%. Deretter gjorde vi tester med både nettverk og klassifisereren, hvorpå vi måtte gjøre mindre endringer i datasettet for å oppnå best mulig resultat. Dette innebar å legge til enkelte rammer som vi tidligere hadde filtrert bort fordi de var for dårlige.

Automatisk gjenkjenning
Vi laget en applikasjon for automatisk gjenkjenning, hvor programmet spør brukeren etter en mappe med videosnutter. Når brukeren har valgt mappe opprettes det en ny katalog som heter "bilder". I denne katalogen vil alle bilderammene fra videosnuttene bli lagt i, videre klassifiserer klassifisereren hvert enkelt bilde i bildekatalogen som ingenfisk, settefisk eller villfisk. Rammene klassifisert som ingenfisk sees bort ifra, og det

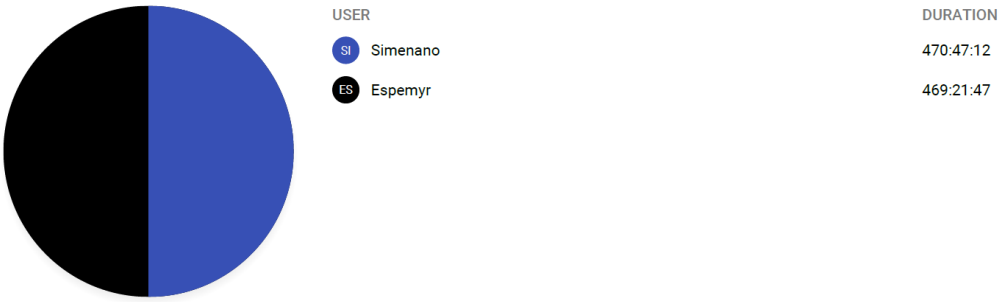
totale resultatet for hver video skrives ut i et Excel-dokument.

7.3 Arbeidsrapport

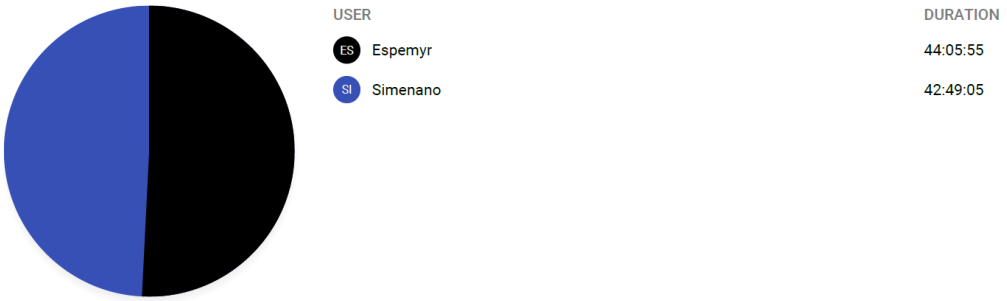
29.april - 20.mai
Den siste tiden var satt av til skriving, der vi kun fokuserte på ferdigstilling av rapport. Dette viste seg å være vanskelig å overholde, så utviklingen fortsatte en drøy uke ut i perioden. Utsettelsen medførte at vi havnet litt bakpå med skrivingen og vi måtte jobbe hardt og lenge for å ta igjen det tapte. Vi hadde kun skrevet stikkord underveis om hva vi skulle ha med på rapporten, så vi måtte skrive mye, og mer utfyllende om hvordan og hva vi hadde arbeidet med. Under skrivingen delte vi ansvar og leste over hva den andre hadde skrevet i rapporten.

7.4 Tidsbruk

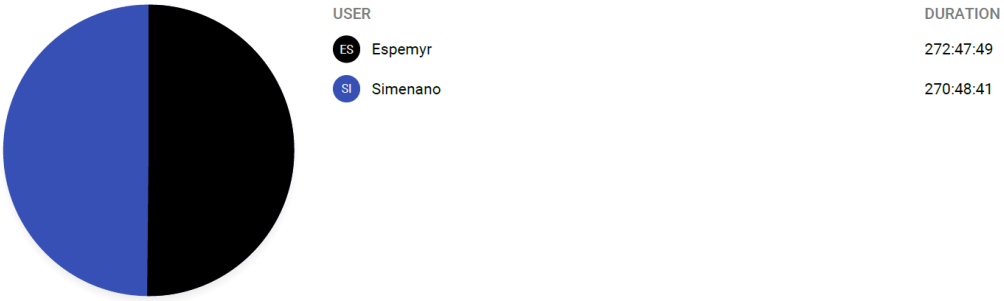
Vi brukte Toggl for å holde styr på tidsbruken under hele prosjektet [2]. Vi var litt slurvete med timeføringen i starten, så enkelte dager mangler. Det var også mange småting underveis hvor vi ikke førte opp tiden, men som sammenlagt ville gjort et utslag på totalen. Kort oppsummert har vi brukt mer tid enn vi så for oss i begynnelsen av prosjektet, men det må vi regne med når det er nytt læringsutbytte. Sammenlagt endte vi på 940:08:59 timer totalt.



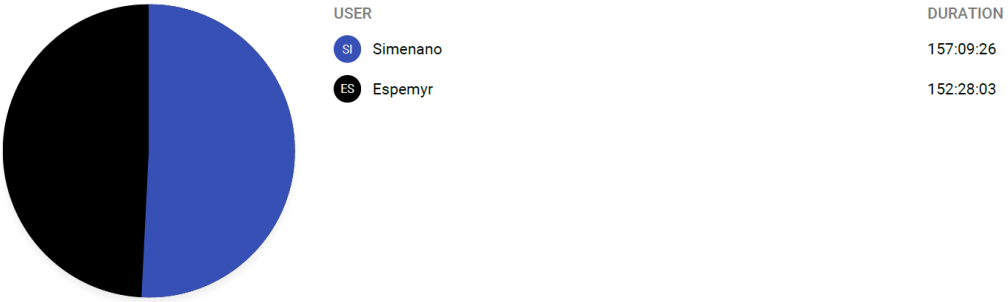
Figur 39: Total tidsbruk bacheloroppgave



(a) Tidsbruk prosjektplan



(b) Tidsbruk prosjekt



(c) Tidsbruk rapport

Figur 40: Tidsbruk i de forskjellige delene av bacheloroppgaven

8 Drøfting

I dette kapittelet drøfter vi arbeidsprosessen, organiseringen og læringsutbyttet av prosjektet i sin helhet.

8.1 Evaluering av gruppearbeidet

Gruppearbeid hadde vi litt erfaring i fra emnene Systemutvikling og Objektorientert Programmering, men ikke på samme nivå som en bacheloroppgave. Vi møttes på presentasjonen av bacheloroppgavene på NTNU sammen med en tredje person som dessverre ikke kunne bli med. Vi hadde interesse for de samme oppgavene, og besluttet å inngå et samarbeid. Vi hadde ikke jobbet sammen før, men vi hadde hatt de fleste av tidligere emner sammen. Vi satte oss ned og bestemte gruppereglene som ligger i vedlegg under prosjektplanen B. Vi snakket også om motivasjon og ønsket resultatoppnåelse. På disse områdene var vi på samme bølgelengde. Da dette var på plass startet arbeidet med å sette seg inn i nytt fagstoff. Vi samarbeidet i starten om å sette oss inn i maskinlæring og datasynteknikker. Dette var en grei måte å sette seg inn i mest mulig på kort tid. På et senere tidspunkt ble det mer aktuelt å dele oss litt mer og fokusere på hver våre oppgaver. Også dette fungerte bra, vi hadde kontinuerlig kontakt med hverandre og holdt hverandre oppdatert. Oppgaven krevde at vi jobbet på kraftig maskinvare, så det aller meste av arbeidet skjedde hjemmefra hvor begge har stasjonære datamaskiner. Det fungerte overraskende bra, selv om det er mer optimalt å jobbe på samme lokasjon under gruppearbeider. Det som derimot ikke fungerte var parprogrammering. Det lot seg ikke gjøre fra forskjellige lokasjoner. Heldigvis finnes det i dag gode samarbeidsverktøy, både for kommunikasjon og skriving, så alt i alt hadde vi et vellykket samarbeid.

8.2 Arbeidsprosess

Arbeidsprosessen startet med at gruppen tilegnet seg nødvendig informasjon for å få forståelse i lærestoffet. Oppdragsgiver hadde ingen spesielle krav eller forutsetninger, slik at vi var fri til å se på ulike teknikker og verktøy for gjennomføring. Hadde oppdragsgiver hatt noen spesielle krav ville mest sannsynlig tiden det tar å tilegne seg læring gått fortere, for eksempel om vi hadde hatt en forutsetning om å lære om et enkelt verktøy istedenfor og se på mange forskjellige og finne fordeler/ulemper. Dette ser vi på som både postivt og negativt, siden arbeidsprosessen vår har vært å sett på flere ulike læringsprosessen slik at vi nå sitter igjen med et større helhetlig bilde av datasyn/maskinlæring.
I starten av prosjektet ble vi anbefalt fra første veiledningsmøte at vi burde dokumentere alt vi gjør og begrunne eventuelle endringer. Dette var vi flinke på i starten de første tre sprintene, men etterhvert når det ble mye forskjellig treninger og testing ble det glemt. Vi fikk kommet fort i gang med selve rapporten fra der vi dokumenterte godt, men når vi kom i de siste to sprintene stoppet det litt opp på grunn av manglende dokumentasjon, men siden det ikke var så lenge siden vi gjennomførte sprinten gikk det heldigvis bra. På slutten av prosjektet ble det veldig mye trening og testing av nettverk. Der ble

det litt uoversiktlig og vi stresset litt med tanke på at vi burde startet med rapportskriving, men vi fikk heldigvis gjort litt skriving mellom kjøringer av nettverk.

8.3 Oppfølging av Scrumban

I utviklingsmodellen hadde vi bestemt oss for å benytte hybriden Scrumban, (seksjon 2.1). Vi hadde også planlagt å benytte verktøyet "Taiga.io" [1], men denne ble frafallende når vi startet på prosjektet. Opprinnelig skulle verktøyet være en indikasjon på hva som var i arbeid og hva som var fullført, men dette benyttet vi oss ikke av. Istedenfor laget vi bare et enkelt dokument i LaTeX hvor vi noterte ned hva som skal gjøres i hver sprint og hva som har blitt gjort. Siden vi ikke hadde noen kunnskap og tidsestimat for hvor lang tid hver enkelt arbeidsoppgave tok, hadde vi ikke planlagt på forhånd hva vi skulle gjøre til hver enkelt sprint. Dette ble planlagt i starten av hver sprint da vi hadde sprint planning møte.

Om vi skulle gjennomført utviklingsmodellen på nytt, burde vi tatt i bruk "Taiga.io", som gir en god oversikt over hvem som har ansvar over arbeidsoppgaver og hvilke arbeidsoppgaver som pågår istedenfor et dokument som kan være litt "rotete". Vi skulle også dokumentert mye hyppigere, selv om vi var gode på dokumentasjonen i starten ble den frafallende på slutten når det blir mange arbeidsoppgaver i forbindelse med forskjellige treninger og tester.

8.4 Organisering

Gruppen ble enige om å møte opp klokken 8 om morgenen for å kunne jobbe felles og ha en strukturert arbeidsflyt med oppgaven. Dette gjorde vi alle hverdager så samt vi ikke hadde noen obligatoriske hendelser vi måtte være med på. Vi hadde en liten dialog på starten av hver dag slik at vi visste hva hverandre jobbet med til enhver tid. Timeloggene ble ført i Toggl [2] slik at vi får et overblikk over hvor mange timer det blir totalt i løpet av oppgavetiden. Vi hadde sprint retrospektivt møte etter hver sprint slik at vi hadde oversikt og dokumenterte hva som har blitt gjort i løpet av sprinten, samtidig hadde vi et sprint planleggings møte for å planlegge videre arbeid for kommende sprint. Veileder Sony George satte opp ukentlig møte, men dette ble justert i forbindelse med progresjonen vår. Ukentlig møte med veileder var en god organisering, da det er lettere å avlyse møte enn å booke et. Vi fikk også holdt veilederene Sony og Marius oppdatert på hva vi jobbet med og fikk tilbakemelding vedrørende resultater og status på hvordan vi lå an.
I starten av prosjektfasen fikk vi spørsmål om vi skulle skrive rapporten på engelsk eller norsk. Sony George kan ikke så godt norsk, så vi måtte gjøre en liten vurdering på hvilket språkbruk vi skal benytte. Vi valgte norsk som språk på grunnlag av egne erfaringer, selv om det letteste med tanke på ord og uttrykk hadde vært å skrevet rapporten på engelsk.
Marius hjalp oss med tilgang på fargelabben på NTNU i Gjøvik midt i semesteret når vi skulle starte å trene hyppigere slik at vi kunne trene nettverk og klassifisering på to datamaskiner, det var veldig nyttig for oss med tanke på tidsbruken. Der kunne vi logge datamaskinen fra hjemme slik at vi kunne sette på kjøringer til enhver tid.
Når vi startet med rapportskriving brukte vi LaTeX som nevnt i seksjon 1.5, dette verk-

tøyet bidro til at vi kunne samarbeide og se hverandre sitt bidrag i sanntid. I tillegg til dette har vi en egen "chat" funksjon der vi kan stille hverandre spørsmål og drøfte ulike problemstillinger underveis i rapportskrivingen.
Organisering av strukturen på rapporten hadde vi noe problem med i starten, da vi skrev ned kronologisk hvordan vi hadde arbeidet med prosjektet, men etter gode innspill fra veileder fikk vi etterhvert en struktur som vi er fornøyde med. Vi tar for oss slik at kapitlene som påvirker eller inneholder teori skal komme før vi gjør noe praktisk uten å snakke generelt om det på kapitlet før.
Vi jobbet en uke ekstra på applikasjonen vår, som gjør at vi fikk dårligere tid på rapportskriving enn det som var opprinnelig planlagt fra prosjektplanen vår. Vi tok avgjørelsen i fellesskap for å gjøre noen ekstra endringer på applikasjonen slik at vi kunne presentere et bedre resultat.

Vi er stort sett fornøyde med organsieringen av prosjektet. Gjennom prosjektperioden har vi vært inkluderende slik at vi har begge forståelse i hva vi gjør og hvorfor ting blir gjort.

8.5 Fordeling av arbeidet

Vi måtte lære oss datasyn og maskinlæring, siden ingen av oss ikke hadde noen erfaring fra tidligere. Dette var en ganske omfattende del av oppgaven, så dette måtte vi begge være inkludert i. Marius ga oss innsyn til emnet "Computer Vision - IMT3017" tidlig i semesteret slik at vi fikk tilgang på nyttige ressurser. Vi fortsatte samme fordeling når vi skulle bearbeide datasettet og preprosessere bilder, slik at vi begge har innblikk i teknikkene som blir brukt og hvordan man tilføyer teknikkene. På slutten når vi skulle starte med applikasjonen fordelte vi ansvaret litt forskjellig, den ene kodet selve treningen av nettverket, mens den andre kodet klassifiseringsapplikasjonen. Selv om noe av arbeidemetodikken var kjent, har det vært mye nytt å lære seg. Som gruppe føler vi at vi begge har vært inkludert og fått en god forståelse i hvordan datasyn og maskinlæring fungerer i praksis, noe som vi kanskje ikke hadde fått om vi hadde vært flere på gruppen og fordelt ansvaret annerledes.

8.6 Læringsutbytte

Læringsutbytte til gruppen har vært stor. Vi har ikke hatt noen tilknytning mot datasyn og/eller maskinlæring fra tidligere, slik at læringsutbyttet vårt ble som å ta et helt nytt emne ved siden av bacheloroppgaven.

Læringsutbyttet gruppen har fått:

- Datasyn
- Maskinlæring
- Matlab
- Python
- Preprosessering av digitale bilder/video
- Akademisk skriving

Vi har også fått innsikt i hvordan det er å jobbe i et lite team, hvor vi har god dialog med veileder som sparringspartner når vi har spørsmål og trenger veiledning i forbindelse med prosjektet. Prosjektarbeidet har gitt oss god lærdom, vi er veldig fornøyde med hvordan vi oppnådde resultatene med tanke på bakgrunnen vi hadde før prosjektstart. Eneste vi kan utsette på blir dialogen med oppdragsgiver, da vi kun fikk tatt et møte via Skype i starten av prosjektet, da det gjerne kunne vært mer samarbeid med oppdragsgiver slik at vi får en følelse av at det er en oppgave for noen istedenfor kun et vanlig prosjekt.

8.7 Kritikk av oppgaven

I forbindelse med kritikk til oppgaven, kan vi ta med at vi syntes den var relativ "åpen" og etter første møte med oppdragsgiver fikk vi bare generell informasjon rundt motivasjonen for oppgaven, men ikke noe spesifikt rundt oppgaveløsningen eller krav/-forutsetninger. Vi brukte litt ekstra tid i starten i forbindelse med å sette egne krav og forutsetninger slik at vi kan løse oppgaven basert på våre egne forutsetninger. Det har jo forsåvidt vært en god læringsprosess å kunne sette egne krav til en oppgave, men det hadde kanskje vært det beste for oss med tanke på vår bakgrunn at det var noen forutsetninger eller krav fra oppdragsgiver i starten.

9 Konklusjon og videre arbeid

9.1 Konklusjon

Gjennom dette prosjektet har vi lært veldig mye nytt om datasyn og bildebehandling, samt maskinlæring. Vi har tatt i bruk flere av dagens mest populære teknikker for augmentering og løsing av klassifiseringsproblemer. Vi har sett viktigheten av en tilstrekkelig mengde data, og vist at et godt balansert datasett har stor innvirkning på balansen i resultatet. Vår egen utførelse har lært oss hvordan et utviklingsprosjekt bør, og til tider ikke bør utføres. Gjennom prosjektet og andres litteratur har vi tilegnet oss kunnskap og en erfaring innen to nye fagområder som vi ikke hadde drømt om før prosjektstart.

Vi har med vår applikasjon vist at det er mulig å få til automatisk gjennkjenning av settefisk og villfisk med en tilfredsstillende nøyaktighet både med KNN og ECOC. Vi tror og håper at vår applikasjon kan tas i bruk av NINA i deres arbeid med å overvåke og opprettholde en sunn fiskebestand i Gudbrandsdalslågen.

9.2 Videre arbeid

Både gruppen og oppdragsgiver NINA har forslag til videre arbeid og utvikling. NINA uttrykte tidlig et ønske om å se på mulighetene for individgjenkjenning av fisk. Vår applikasjon trenger kun å se på omrisset av fisken for å gjøre klassifiseringen av settefisk og villfisk, ved individgjenkjenning kreves derimot en grundig analyse av fiskens mønster. Dette mener vi at krever en god del klarere bilder enn det vi har sett fra de lavoppløste undervannsfilmene.

Det kan være interessant å overføre applikasjonen til andre fisketyper eller dyr. Det eneste som begrenser hva man kan klassifisere ved bruk av dyp læring slik vi ser det er datasettet. Dette må være balansert og stort nok til at egenskapene ved hver klasse kan læres av modellen.

Robustheten til applikasjonen er testet mot komprimering ved konvertering til JPEG/JPG, men mange andre områder må testes. Spørsmål vi ser på som viktige å svare på, og evt. utbedre er:

- Hva skjer om to fisk fra ulike klasser er med i samme video?
- Hva skjer om noe annet enn en fisk kommer gjennom fisketelleren?
- Hva skjer om en annen art enn Ørret går gjennom fisketelleren?

Til slutt tar vi med endringer på modellene og applikasjonen. Vi har sett at det alltid er rom for forbedring, og det er flere teknikker og løsninger vi ikke fikk tid til å teste:

- Klassifisering av filmklipp direkte, uten å lagre bilderammer. Dette rakk vi ikke å se på, men kan være verdt å utforske videre.
- Utviklingen av KNN har skutt fart de siste årene, og det er flere nettverk vi gjerne skulle testet, som enten ikke er tilgjengelig i Matlab, eller som tok for lang tid å trene.
- Oversetting av applikasjonen til f.eks. Python. Matlab er som vi har erfart veldig godt og enkelt å jobbe i, men er mindre fokusert på å rulle ut og distribuere applikasjoner. Derfor tenker vi det kan være mye å hente med tanke på optimalisering ved å oversette applikasjonen.

Bibliografi

[1] Taiga. (2019). Taiga io, side: https://taiga.io (Sist besøkt 19. jan. 2019).

[2] Toggl. (2019). Toggl, side: https://toggl.com/ (Sist besøkt 19. jan. 2019).

[3] Google. (2019). Google Drive, side: https://www.google.com/drive/using-drive/ (Sist besøkt 19. jan. 2019).

[4] BitBucket. (2019). BitBucket, side: https://bitbucket.org/ (Sist besøkt 25. jan. 2019).

[5] S. George, *IMT3017 - Introduction to CV.* 2018.

[6] Techpin. (2019). How does a digital camera work?, side: https://techpin.com/who-invented-the-digital-camera/ (Sist besøkt 16. mai 2019).

[7] W. contributors. (2019). MPEG — Wikipedia, The Free Encyclopedia, side: https://no.wikipedia.org/wiki/MPEG (Sist besøkt 18. mai 2019).

[8] M. Pedersen, "Image quality course", 2018.

[9] M. Silvetti og T. Verguts, "Reinforcement learning, high-level cognition, and the human brain", i *Neuroimaging-Cognitive and Clinical Neuroscience*, IntechOpen, 2012.

[10] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel mfl., "Mastering chess and shogi by self-play with a general reinforcement learning algorithm", *arXiv preprint arXiv:1712.01815*, 2017.

[11] O. helse. (2019). Nervecelle, side: https://omhelse.no/wp-content/uploads/2013/07/Nervecelle.jpg (Sist besøkt 3. mai 2019).

[12] D. Tanikić og V. Despotovic, "Artificial intelligence techniques for modelling of temperature in the metal cutting process", i *Metallurgy-Advances in Materials and Processes*, IntechOpen, 2012.

[13] V. Dumoulin og F. Visin, "A guide to convolution arithmetic for deep learning", *arXiv preprint arXiv:1603.07285*, 2016.

[14] D. M. Hawkins, "The problem of overfitting", *Journal of chemical information and computer sciences*, årg. 44, nr. 1, s. 1–12, 2004.

[15] A. Karpathy. (). Convolutional Neural Networks for Visual Recognition.

[16] J. Brownlee. (). A Gentle Introduction to Transfer Learning for Deep Learning.

[17] A. Krizhevsky, I. Sutskever og G. E. Hinton, "Imagenet classification with deep convolutional neural networks", i *Advances in neural information processing systems*, 2012, s. 1097–1105.

[18] Medium. (2019). CNN Architectures—LeNet, AlexNet, VGG, GoogLeNet and ResNet, side: https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848 (Sist besøkt 7. mai 2019).

[19] ML-cheatsheet. (2019). Activation Functions, side: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html (Sist besøkt 8. mai 2019).

[20] T. Pengying, "Underwater Fish Classification based on Alexnet", 2018.

[21] S. Ruder, "An overview of gradient descent optimization algorithms", *arXiv preprint arXiv:1609.04747*, 2016.

[22] A. Agarwal. (2019). JPEG or PNG – Which Image Format Offers Better Quality?, side: https://www.labnol.org/software/tutorials/jpeg-vs-png-image-quality-or-bandwidth/5385/ (Sist besøkt 3. mai 2019).

[23] E. Junqué de Fortuny, D. Martens og F. Provost, "Predictive modeling with big data: is bigger really better?", *Big Data*, årg. 1, nr. 4, s. 215–226, 2013.

[24] R. Kumar Rai, P. Gour og B. Singh, "Underwater image segmentation using clahe enhancement and thresholding", *International Journal of Emerging Technology and Advanced Engineering*, årg. 2, nr. 1, s. 118–123, 2012.

[25] Mathworks. (2019). Extract Image Features Using Pretrained Network, side: https://se.mathworks.com/help/deeplearning/examples/extract-image-features-using-pretrained-network.html (Sist besøkt 8. mai 2019).

[26] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro og B. Recht, "The marginal value of adaptive gradient methods in machine learning", i *Advances in Neural Information Processing Systems*, 2017, s. 4148–4158.

[27] G. E. Sakr, M. Mokbel, A. Darwich, M. N. Khneisser og A. Hadi, "Comparing deep learning and support vector machines for autonomous waste sorting", i *2016 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*, IEEE, 2016, s. 207–212.

[28] Mathworks. (2019). trainingOptions, side: https://se.mathworks.com/help/deeplearning/ref/trainingoptions.html (Sist besøkt 18. mai 2019).

[29] T. Dettmers. (). A Full Hardware Guide to Deep Learning.

[30] S. Dodge og L. Karam, "Understanding how image quality affects deep neural networks", i *2016 eighth international conference on quality of multimedia experience (QoMEX)*, IEEE, 2016, s. 1–6.

VEDLEGG

A Møtereferat

Referat veiledningsmøte
Dato:8.Januar 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 1
Beskrivelse
Gruppen møtte Sony (veileder) og Marius for å få starthjelp i prosjektet. Det ble drøftet hvordan vi skulle gå i gang med oppgaven og forskjellige tips rundt gjennomføring av oppgaven. Nedenfor er det en del punker om hva vi gjennomgikk:

• **Hva oppgaven går ut på**
Finne fisk med manglende finne.

• **Metodebruk for gjennomføring av oppgaven (datasyn)**

• **Datasett**
Hvor mye data vi får tilgjengelig.

• **Bilder vs video**
Video er mer avansert, skille ut bilder fra videosnutter.

• **Hvorfor vi valgte det, og hvilke krav NINA har til oss**

• **Tidsplan**
Bruke Gantt skjema, planlegge frem i tid, sette milepæler.

• Møte med veileder - ukentlig

• **Ambisjoner**
Hvilken karakter vi ønsker oss.

•**Språkbruk**
Skrive bachelor på engelsk eller norsk?

• **Verktøy**
Hvilke verktøy trenger vi? Git,Trello,Latex,Jabref.

• **Verktøy**
Skrive ned hva vi kunne fra før, hvem gjør hva.

• **Ressurser**
Bruke ressursene vi har tilgjengelig på skolen, veiledere og PhD studenter, litteratur.

Referat veiledningsmøte
Dato:14.Januar 2019
Sted:NTNU i Gjøvik
Emne: Veiledningsmøte 2
Beskrivelse
Gjennomgikk spørsmålene vi har skrevet ned i forbindelse med møte med oppdragsgiver, veileder kom også med innspill til hva vi kan stille til oppdragsgiver.

Referat veiledningsmøte
Dato:28.Januar 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 3
Beskrivelse
Vi gjennomgikk naturlige grenser og krav i forbindelsen med oppgaven da oppdragsgiver ikke har noen spesielle krav til oss. Noen av kravene vi ble enige om:

- Kun ørret som går gjennom fisketrappen på Hunderfossen
- Kun en fisk om gangen i videosnutten
- Velge gode bilder fra videosnuttene i starten.

Referat møte med NINA
Dato:29.Januar 2019
Sted: NTNU i Gjøvik
Emne: Første møte med oppdragsgiver, signere prosjektavtale.
Beskrivelse
Oppdragsgiver hadde ikke mulighet til å komme til Gjøvik og møtet ble gjennomført via Skype.
Vi sendte over prosjektavtalen på e-post og fikk tilbake ferdigsignert avtale av Jon.
Vi hadde satt opp en liste med forskjellige spørsmål til oppdragsgiver. Noen av spørsmålene vi hadde

- Krav/ønsker til oss
- Entydig beskrivelse av mål og problemstilling
- Hvorfor vi gjør dette, hva er motivasjonen.
- Hvor mye data vi får tilgjengelig
- Krav til software/teknologi
- Hvor stor treffsikkerhet skal gjennkjenningen være.

- Hvor lang tid de bruker på dette manuelt
- Krav til bruker

Referat veiledningsmøte
Dato:12.Februar 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 4
Beskrivelse
Vi hadde et møte om forskjellige datasyn teknikker, da vi var i en periode med egenlæring, hvor vi hadde noen spørsmål rundt bruk av teknikker. Vi snakket om forskjellene mellom OpenCV og Matlab, der vi ble mer henvist mot Matlab grunnet veilederene sin kompetanse. Vi ble oppfordret til å se mer på tradisjonell maskinlæring og dyp læring.

Referat veiledningsmøte
Dato:5.Mars 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 5
Beskrivelse
Vi hadde fått tilgang på datasettet fra oppdragsgiver og startet med å bearbeide dette. Vi viste frem datasettet og at vi hadde konvertert fra video til bilde. Fikk flere tips fra Marius om å blant annet konvertere til PNG-format istedenfor JPG-format da man kan miste viktig detaljer i bilder grunnet kompresjon. Gikk til slutt gjennom forskjellige teknikker som kan forsterke eller utvide datasettet vårt.

Referat veiledningsmøte
Dato:18.Mars 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 6
Beskrivelse
Dette veiledningsmøtet startet vi med å vise frem de ulike teknikkene vi har påført datasettet vårt, videre startet vi med å diskutere rundt ferdigtrente nettverk, da det var på tide starte med selve applikasjonen som har milepæl den 1.April. Vi ble anbefalt teste ferdige applikasjoner for å få innsikt i hvordan man behandler ferdigtrente nettverk og hvordan Matlab fungerer.

Referat veiledningsmøte
Dato:25.Mars 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 7
Beskrivelse
Startet møte ved å vise frem resultat fra ulike tester vi har gjort.

Nøyaktighet i %	SGDM	Adam	RMSProp
Uten CLAHE	94.06%	91.33%	81.20%
Med CLAHE	91.46%	93.07%	86.26%

Tabell 1: Nøyaktighet av læringsalgoritmer

Vi fikk et par tilbakemeldinger fra Marius vedrørende testing:

- Kjøre læringsalgoritmene flere ganger og ta gjennomsnittet, se om nøyaktigheten er stabil.
- Fjerne bilder med halve fisken, trene på kun hel fisk?

Referat veiledningsmøte
Dato:3.April 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 8
Beskrivelse
Vi gjennomgikk struktur på rapporten, lage innholdsfortegnelse og se hvordan vi skal presentere rekkefølgen på en god måte. Veileder ønsker også at vi skal opprette en tabell over bilder som blir klassifisert feil, slik at vi kan vise dette i rapporten og ta med i den muntlige presentasjonen.

Referat veiledningsmøte
Dato:12.April 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 9
Beskrivelse
Dette veildeningsmøtet drøftet presentasjon av rapporten.
Noen av punktene vi drøftet:

- Hvilke resultater skal inn i rapporten?
- Analysere feilklassifisering
- Presentere det på en måte som gir mening for leseren
- Presentere kronologisk eller gruppere etter tema?
- Diskutere klassifisering av bilder vs video
- Hvordan formidle det vi har gjort på en god måte?

Referat veiledningsmøte
Dato:23.April 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 10
Beskrivelse
Startet med å gjennomgå resultater av bilder som blir klassifisert feil.Veilder ba oss opprette et histogram for å vise frem nøyaktigheten til hver enkelt videosnutt. Diskuterte rundt kvaliteten på bildene og hvordan vi skal løse problemet med at vi får overskuddsbilder ved kjøring av hele videosnutter.

Referat veiledningsmøte
Dato:2.Mai 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 11
Beskrivelse
Vi hadde satt opp noen spørsmål på forhånd i forbindelse med innholdsfortegnelsen og presentasjonen i rapporten. Fikk gode tilbakemeldinger på presenteringen og hva som er viktig å ta med i rapporten for å vise at vi har testet forskjellige teknikker. Ta med fremtidig arbeid, hva som kan gjøres bedre og teste enda mer med forskjellige teknikker, kan bruke koden til å identifisere andre fiskeslag/dyr.

Referat veiledningsmøte
Dato:16.Mai 2019
Sted: NTNU i Gjøvik
Emne: Veiledningsmøte 12
Beskrivelse
Vi hadde satt opp noen spørsmål på forhånd i forbindelse med ferdigstilling av rapporten. Så gjennom strukturen og endret på rekkefølgen slik at den gir mer mening i forbindelse med strukturen og at leseren skjønner hva vi har arbeidet med.

B Prosjektplan

NTNU

Norwegian University of
Science and Technology

Automatisk gjenkjenning av settefisk
Prosjektplan

Espen Myrum, Simen Nørstebø

19. mai 2019

Innhold

Figurer

1 MÅL OG RAMMER

1.1 Bakgrunn

Bachelor i ingeniørfag data på Gjøvik avsluttes med bacheloroppgaven. Denne oppgaven skal være forankret i reelle problemstillinger fra samfunns- og næringsliv eller forsknings- og utviklingsarbeid og bidra til innføring i vitenskapsteori og metode, samt integrere viktige deler av programmets faglige innhold. Prosjektet skal bunne ut i en rapport som dokumenterer dette arbeidet på en strukturert måte. I den forbindelse skal vi utføre et prosjekt for Norsk Institutt for Naturforskning (NINA). NINA er en uavhengig stiftelse som forsker på natur og samspillet natur - samfunn.

Hunderfossen kraftverk stod ferdig i 1964. Det ble etablert ei fisketrapp for å få gytefisk av ørret forbi den store kraftverksdemningen og det ble også etablert ei fysisk felle midt i fisketrappa slik at man kunne overvåke fiskeoppgangen. Dette har gitt viktige data om utviklingen i ørretbestanden, men samtidig har undersøkelser vist at fangst av ørret i fella og håndtering av fisken (bl.a. måling og merking) kan ha negative konsekvenser for fisken. Det har vært et ønske om å raffinere overvåkingsmetoden og i 2016 ble det installert en automatisk fisketeller slik at fisken kan svømme fritt gjennom fisketrappa. Hos all kultivert ørret fra Hunderfossen er "fettfinna" klipt av før utsetting som 2-åringer. Etter noen år i Mjøsa kommer disse tilbake som gytefisk til Gudbrandsdalslågen. Det er enkelt å skille naturlig rekruttert ørret (villfisk) fra kultivert ørret (settefisk) hvis man går igjennom alle filmene manuelt. Hver fisk blir registrert med et "scannerbilde" og en filmsnutt på ca. 15 sekunder. Dette er viktige data fordi man ønsker å øke den naturlige produksjonen av villfisk i elva.

1.2 Prosjektmål

1.2.1 Resultatmål

Målet for bacheloroppgaven er å implementere datasynteknikker og eventuelt maskinlæring for å skille naturlig rekruttert ørret og kultivert ørret ut i fra film fra fisketrappa i Hunderfossen kraftverk i Gudbrandsdalslågen.

1.2.2 Effektmål

NINA stiller få spesifikke krav til effektmål, men hensikten vil først og fremst være en reduksjon i tiden som nå brukes på gjennkjenning og telling av settefisk og villfisk. NINA estimerer tidsbruken i dag til ca. to ukesverk, eller rundt 80 timer, på å gå gjennom ett års innsamlede videoer. Dette med utgangspunkt i telleren ved Hunderfossen og de rundt 1000 fiskene som passerer denne i løpet av ett år. Ved å automatisere prosessen håper vi å kunne spare inn en god del av denne tiden.

For at dette skal være effektivt nok til å implementeres i dagens system ønsker NINA en nøyaktighet på minst 90 prosent.

Manuelle feller har vist å påføre fisken stress, og brukes ifølge NINA fortsatt flere steder til merking og måling. Hvis vår løsning viser seg å forminske ressursbehovet vil det være ønskelig å innføre et liknende system også andre steder.

1.3 Rammer

Prosjektet har en tidsramme fra 7.januar til 20. mai.

Hvilke rammer oppdragsgiver har til oss:

• Ingen bestemte rammer på hvilken programvare/teknologi vi skal ta i bruk, vi står derfor fritt til å velge dette selv.

• Ingen strenge krav til leveransen, oppdragsgiver ønsker å se hva vi klarer å få til med datasettet vi får utdelt.

1.4 Ressursbehov

Hvis vi velger å gå for maskinlæring som teknikk for å kunne utføre oppgaven har vi behov for utstyr som takler dette. For å gjøre effektiv maskinlæring trenger vi for eksempel god grafikkprosessor som gir god ytelse. Fargelaben på NTNU i Gjøvik har maskiner som takler dette, noe vi forhåpentligvis kan få tilgang til.
Ved bruk av maskinlæring trenger vi også tilgang på mye data fra NINA. Dersom vi får tilgang på et begrenset antall videoklipp må vi eventuelt lene oss mer mot en annen retning. Vi har etter samtaler med oppdragsgiver fått tilbud om hele NINAs datasett.

2 OMFANG

2.1 Fagområde

Oppgaven gir oss frihet til å benytte datasyn og maskinlæring for å lage en programvare som filtrerer om det er kultivert ørret eller villfisk. I den forbindelse kreves det kunnskap i en del ukjente programmeringsspråk. I oppstartsfasen vil vi se nærmere på hvilke teknikker som gir best utbytte for oss i vårt prosjekt. Vi vil også se på de mest populære programmeringsspråkene brukt innen datasyn, MATLAB og Python, før vi avgjør hva vi skal bruke. Vi skal skille ut bilder fra datasettet vi blir tildelt og bygge videre på dette for prosjektet vårt.

2.2 Avgrensning

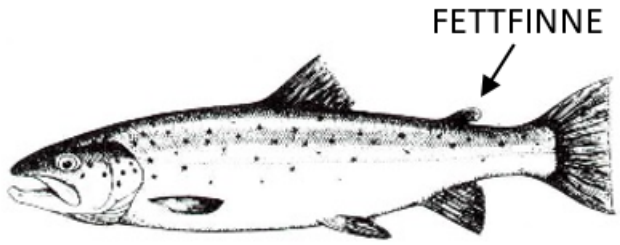
Vi ble nødt til å avgrense prosjektet noe da vi opprinnelig skulle være tre medlemmer, men etter en uforutsett hindring endte vi opp med å bli kun to.

Oppdragsgiver ønsket å se om det var mulig gjenkjenne samme ørreten som var i fiskefrappen om den kom tilbake noen år senere. Vi ble enige sammen med oppdragsgiver at det ble for krevende og at vi kun skal fokusere på gjenkjenning mellom kultivert ørret og villfisk.

Ingen av medlemmene i gruppen har tidligere erfaring med datasyn, maskinlæring og/eller andre teknikker, så det vil bli satt av tid i forkant av selve utviklingsperioden for å lære dette.

2.3 Oppgavebeskrivelse

I mange elver og vann i Norge brukes settefisk for å opprette fiskebestander. Settefisk er kunstig klekkede fiskeunger som fôres en tid før de settes ut i ferskvann eller sjøen for videre vekst. Før disse settes ut så fjernes fettfinnen på fisken, dette gjøres for å kunne gjenkjenne settefisk fra villfisk. Norsk Institutt for Naturforskning (NINA) har montert kamera i flere norske elver for å kunne overvåke fiskebestander. I dag er gjenkjenningen av settefisk en manuell oppgave. I denne oppgaven ønskes det å bygge en automatisk metode for gjenkjenning av settefisk. Gjenkjenning baseres på undervannsbilder (video). Disse bildene er tatt under forskjellige forhold (forskjellige lysforhold, farge på vannet, luftbobler, etc.). Gjenkjenningen må være robust og fungere under alle slags forhold.



Figur 1: Ørret med fettfinne. [1]

2.3.1 Krav

Oppdragsgiver har ingen spesifikke krav til oppgaven, vi har derfor satt opp våre egne.

• Videoklippene skal kun være ørret og sortert etter settefisk/villfisk av oppdragsgiver.

• Videoklippene skal kun vise en ørret om gangen som går gjennom fisketrappen i Hunderfossen.

• Utviklingsmodellen skal være tilpasningsdyktig slik at vi kan legge til funksjonalitet dersom vi får tid til dette.

• Tilgang på Fargelaben på NTNU i Gjøvik

2.3.2 Ønsker

Oppdragsgiver har ingen spesifikke ønsker til oppgaven, vi har derfor satt opp våre egne.

• Vi får tildelt så mye data som vi har behov for.

3 PROSJEKTORGANISERING

3.1 Ansvarsforhold og roller

Vi har følgende roller for vårt prosjekt:

Interne roller:	
Gruppeleder	Espen Myrum
Kommunikasjonsansvarlig	Simen Nørstebø
Eksterne roller:	
Oppdragsgivers kontaktperson	Jon Museth
Veileder	Sony George

4 PLANLEGGING, OPPFØLGING OG RAPPORTERING

4.1 Alternativer til utviklingsmodell

Vi har vurdert modellene Scrum, Kanban, Scrumban og Parprogrammering for utvikling av vårt prosjekt.

Scrum er en smidig utviklingsmodell, den tilbyr en gitt struktur med sprinter. Sprinter deles igjen inn i mindre oppgaver, og har et fast intervall som bestemmes ved oppstart. Selve metoden krever mye arbeid med organisering og dokumentering, da vi er to medlemmer vil dette ta mye tid og vi trenger den arbeidskraften vi har til rådighet.[2]

Kanban er en enkel modell som benytter Kanban Board hvor arbeidet blir delt opp og organisert i forskjellige tilstander. På denne tavlen vil vi få en god oversikt over hva som må gjøres og hva som har blitt gjort.[3]

Scrumban er en hybrid av Scrum og Kanban. Denne modellen gir frihet til å velge enkelte artefakter og proseser fra hver enkelt modell. Dermed kan man benytte smidigheten til Scrum, og samtidig dra nytte av arbeidsmetodikken i Kanban.[4]

Parprogrammering er en smidig modell som baserer seg på to programmerere som deler en enkelt datamaskin. Programmereren på tastaturet kalles vanligvis føreren", den andre er aktivt involvert i programmeringen, men fokuserer mer på den totale utførelsen som kalles navigatøren"[5].

4.2 Valg av utviklingsmodell

Ved valg av utviklingsmodell er det flere argumenter som er med på avgjørelsen vår.

- Vi er en liten gruppe på to medlemmer.
- Vanskelig å tidsestimere utfordringer i prosjektet, smidig utvikling vil være en en fordel.
- Prosjektet har ingen sluttilstand, det er mulig å tilføye og utvikle mer.

Valget vårt falt på Scrumban for utvikling av prosjektet. Siden det er en hybrid av Scrum og Kanban gir det prosjektet vårt fleksibilitet til å tilpasse seg og gjøre endringer underveis om vi får tid til å legge til mer funksjonalitet, eller om det oppstår uforutsette utfordringer.

Denne modellen gir oss struktur og prosesser fra Scrum og den åpenheten som Kanban tilbyr. Kanban tilbyr oss Kanban Board som gir oss lett oversikt over vår egen utvikling, om hva som har blitt gjort og hva som skal gjøres. Vi har tenkt å benytte "Taiga.io"[6] som er et verktøy hvor vi kan jobbe med både Scrum sprinter og Kanban Board. Parprogrammering lar oss begge følge med på hva som blir gjort, når det blir gjort, samt at vi kan gi hverandre innspill under utviklingen. Dermed trenger vi ikke å vente helt til gjennomgangen av det som har blitt gjort med å gjøre mindre endringer, eller foreslå forbedringer.

Scrum vs Kanban vs Scrumban

	SCRUM	KANBAN	SCRUMBAN
Board / Artifacts	simple board product backlog sprint backlog product increment burndown chart	mapped on the process board	mapped on the process board
Ceremonies	daily scrum sprint planning sprint review sprint retrospective	none required	daily scrum other scrum related ceremonies IF needed
Prioritization	Part of backlog grooming, done by PO.	Out of the process. Backlog should be prioritized.	Out of the process. Backlog should be prioritized.
Who feeds the work in progress ("brings new work")?	PO	Depends on defined roles and necessities	Depends on defined roles and necessities
Iterations	yes (sprints)	no (continuous flow)	not mandatory (continuous flow); could have sprints

Figur 2: Scrum vs Kanban vs Scrumban [7]

4.3 Plan for statusmøter

Fra Scrum velger vi å bruke sprint retrospective møte etter hver sprint, der vi skal se over det som har blitt gjort i løpet av arbeidsuken. Etter dette vil vi ha et sprint planning møte for å planlegge hva som skjer neste sprint. Det er også satt opp ukentlig møte med veileder, men dette vil justeres etter behov. I samråd med oppdragsgiver skal vi ha tre møter, men dato og tid for dette er ikke satt.

5 ORGANISERING AV KVALITETSSIKRING

5.1 Dokumentasjon, standardbruk og kildekode

Alle avgjørelser, diskusjoner og punkter som tas opp under møter med veileder og/eller oppdragsgiver skal dokumenteres på en god måte og legges inn i gruppens Google Drive[8]. Det samme skal alle andre dokumenter som opprettes eller brukes i forbindelse med prosjektet.

For rapporter benytter vi LaTeX som gir et stort sett med funksjoner. Til gjengjeld krever det mye innsats for å sette seg inn det.

Utviklingsmiljø og programmeringsspråk vil bli valgt etter vi har vurdert ulike teknikker, da det avhenger i stor grad av dette.

For kildekode skal vi benytte Bitbucket[9] da dette tillater oss å jobbe fra eksterne lokasjoner i perioder hvor vi ikke benytter parprogrammering, samtidig som det gir en ekstra forsikring mot tap av data som er lastet opp. Det sikrer også at begge til enhver tid har siste versjon av koden.

I koden vil vi benytte beste praksis med tanke på kommentering, slik at leseligheten er god samt at det som utføres er lettoppfattelig. Det vil si at vi kommenterer idet vi skriver koden, da dette er både utfordrende og tidkrevende i ettertid.

5.2 Risikoanalyse

Risikomatrise					
Sannsynlighet	Svært sannsynlig	Moderat	Høy	Kritisk	Kritisk
	Sannsynlig	Lav	Moderat	Høy	Kritisk
	Lite sannsynlig	Lav	Lav	Moderat	Høy
	Usannsynlig	Lav	Lav	Lav	Moderat
		Ubetydelig	Betydelig	Alvorlig	Svært alvorlig
	Konsekvens				

Figur 3: Risikomatrise

5.2.1 Beskrivelse av risikomatrisen

Vi har i vår risikomatrise valgt å bruke fire nivåer av sannsynlighet, fra usannsynlig til svært sannsynlig. Det samme har vi gjort for konsekvens, som rangeres fra ubetydelig til svært alvorlig. Tilsvarende ender dette opp i fire nivåer av risiko, fra lav til kritisk på øverste nivå. Ved å multiplisere sannsynlighet med konsekvens ser man enkelt ut fra matrisen hvor risikonivået ligger. Dette danner grunnlaget for risikoanalysen vår.

Hendelse	Sannsynlighet	Konsekvens	Risiko
Sykdom blant prosjektmedlemmer	Sannsynlig	Alvorlig	Høy
Prosjektmedlem forlater prosjektet grunnet sykdom skade, sykdom eller andre årsaker	Usannsynlig	Svært Alvorlig	Moderat
Får ikke tilgang til videomaterialet	Usannsynlig	Svært Alvorlig	Moderat
Klarer ikke fullføre prosjektet innen gitt tidsramme	Sannsynlig	Svært Alvorlig	Kritisk
Videomateriale vi får er av for dårlig kvalitet	Sannsynlig	Alvorlig	Høy
Uenighet i gruppa ved avgjørelser	Sannsynlig	Ubetydelig	Lav
Tap av kildekode av forskjellige grunner	Lite sannsynlig	Svært Alvorlig	Høy
Får ikke tilgang til nok videomateriale	Lite sannsynlig	Alvorlig	Moderat

Figur 4: Risikovurdering

5.2.2 Beskrivelse av risikovurderingen

Vi har kommet opp med åtte hendelser vi ser som svært relevante for vårt prosjekt. For hver av disse har vi identifisert sannsynligheten og konsekvensen om disse skulle inntreffe. Videre har vi ved bruk av risikomatrisen kommet fram til risikoen for den enkelte hendelsen.

Hendelse	Tiltak
Sykdom blant prosjektmedlemmer	Påse at medlemmene får nok søvn, mat og drikke under prosjektets varighet. Ved sykdom under parprogrammering fortsetter arbeidet alene, og felles gjennomgang av koden vil skje så fort medlemmet er tilbake.
Prosjektmedlem forlater prosjektet grunnet sykdom skade, sykdom eller andre årsaker	Da vi allerede har mistet ett medlem og endte opp som en gruppe på to, er det kritisk om én til måtte forlate prosjektet. I samtale med veileder, samt programansvarlig må vi se på om det er mulig å ytterligere redusere omfanget av oppgaven.
Får ikke tilgang til videomaterialet	Uten videomaterialet har vi ingen oppgave, så tilgang til dette er kritisk for prosjektets gang. Instituttleder Marius Pedersen har fått noe data i forbindelse med utlysning av oppgaven, dette kan vi bruke hvis det tar for lang tid å få direkte fra NINA.
Klarer ikke fullføre prosjektet innen gitt tidsramme	Følge utviklingsmetoden og integrere denne på en strukturert måte, samt god planlegging og estimering av arbeidet. Funksjonaliteten vil implementeres i moduler, slik at vi først får grunnfunksjonaliteten opp og går før vi fokuserer på nye funksjoner.
Videomaterialet vi får er av for dårlig kvalitet	Bildebehandlingsverktøy må brukes for å forsterke kontraster, fjerne uønskede elementer, samt redusere støy.
Uenighet i gruppa ved avgjørelser	Da vi er bare to medlemmer må vi ved større uenigheter ta dette opp med veileder slik at vi får et tredje synspunkt, og velger den beste løsningen for alle parter. Hvis begge løsninger er likeverdige kan avgjørelsen tas ved tilfeldig trekning/myntkast.
Tap av kildekode av forskjellige grunner	Ha gode rutiner for lagring og oppbevaring av data. Alt skal lagres både lokalt og i Github/Bitbucket, slik at vi ved systemkrasj eller andre uforutsette hendelser mister minimalt med arbeid.
Får ikke tilgang til nok videomateriale	Det er flere metoder for å utvide et datasett bestående av bilder. For eksempel ved å rotere, skalere, vende eller tilføre et bilde støy kan man mangedoble datasettet. I verste fall må vi gå bort fra maskinlæring og bruke enklere datasynteknikker.

Figur 5: Risikotiltak

6 PLAN FOR GJENNOMFØRING

I starten av utviklingsprosessen planlegger vi å samle kunnskap om stoff vi ikke er kjent med og et oppstartsmøte med oppdragsgiver. Når kunnskapen er sanket vil vi lage en "pseudokode" om hvordan vi vil løse oppgaven med datasyn, her vil veileder være en god sparringspartner. Videre vil vi arbeide med datasettet vi har blitt utdelt og samler de beste bildene fra videoklippene som vi bearbeider i analysen. Vi er forbedredt på at det vil forekomme utfordringer og komplikasjoner i forbindelse med prosjektet, men dette er vi klare til å ta på strak arm.

6.1 Inkrementer og milepæler

Forprosjekt (01.02.2019)
Prosjektplan, grupperegler og prosjektavtale være levert inn.

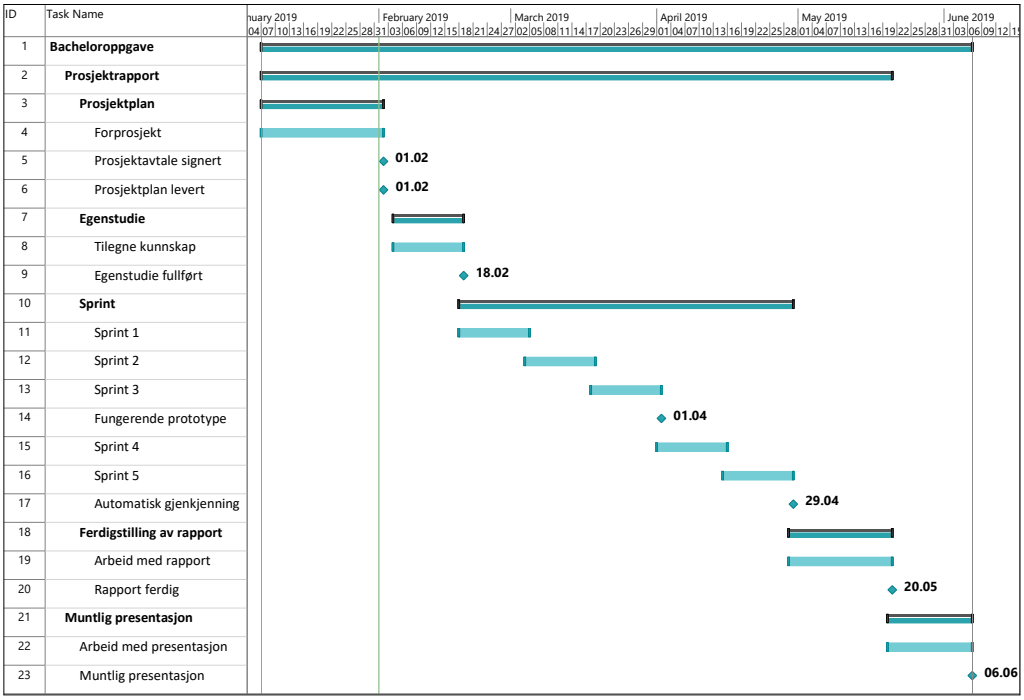
Egenstudie fullført (18.02.2019)
Tilegnet oss nødvendig kunnskap for fortsettelse av prosjekt.

Fungerende prototype (01.04.2019)
Skal ha en funksjonibel prototype som klarer å skille forskjellen på en kultivert ørret/villfisk på grunnlag av bildene vi har tatt fra videoklippene.

Automatisk gjenkjenning (29.04.2019)
Vi skal ha utvilket en programvare som gjenkjenner om fisken er en kultivert ørret eller en villfisk. Resten av prosjekttiden vil bli brukt til ferdigstilling av rapporten.

Innlevering av prosjekt (20.05.2019)
Frist for innlevering av rapporten.

6.2 GANTT



Figur 6: Gantt-skjema

Referanser

[1] SurveyMonkey. (2019). Fangstrapportering for dreggefiskere på Mjøsa, side: https://no.surveymonkey.com/r/mjosa (Sist besøkt 29. jan. 2019).

[2] W. contributors. (2019). Scrum (software development) — Wikipedia, The Free Encyclopedia, side: https://en.wikipedia.org/wiki/Scrum_(software_development) (Sist besøkt 29. jan. 2019).

[3] ——, (2019). Kanban (development) — Wikipedia, The Free Encyclopedia, side: https://en.wikipedia.org/wiki/Kanban_(development) (Sist besøkt 29. jan. 2019).

[4] LeanKit. (2019). What is Scrumban?, side: https://leankit.com/learn/agile/what-is-scrumban/ (Sist besøkt 29. jan. 2019).

[5] W. contributors. (2019). Pair programming — Wikipedia, The Free Encyclopedia, side: https://en.wikipedia.org/wiki/Pair_programming (Sist besøkt 29. jan. 2019).

[6] Taiga. (2019). Taiga io, side: https://taiga.io (Sist besøkt 19. jan. 2019).

[7] T. N. Van. (2019). Scrum Process For Offshore Team, side: https://www.slideshare.net/thiennv1211/scrum-process-for-offshore-team (Sist besøkt 29. jan. 2019).

[8] Google. (2019). Google Drive, side: https://www.google.com/drive/using-drive/ (Sist besøkt 19. jan. 2019).

[9] BitBucket. (2019). BitBucket, side: https://bitbucket.org/ (Sist besøkt 25. jan. 2019).

7 VEDLEGG

Grupperegler

Espen Myrum, Simen Nørstebø

11. Januar 2019

1 Introduksjon

Grupperegler for utføring av bachelor prosjekt. Den skal leses og undertegnes av gruppens medlemmer.

2 Krav

- Hvert gruppemedlem skal legge minst 25 timer i uka på prosjektet, men vi tar hensyn til at tidsbruken kan variere fra uke til uke. Det må tas høyde for økt antall timer ved hektiske perioder i prosjektet.
- Hovedsaklig vil arbeidstiden være fra hverdager 08:00 til 16:00. Arbeid utover dette kan ikke forventes av alle da noen har deltidsjobb på kveldene.
- Arbeidstid skal daglig føres inn i excel ark som ligger i google drive, det skal bli ført antall timer og en liten oppsummering/stikkord om hva som har blitt gjort.
- Møte opp på oppsatte aktiviteter. Dette inkluderer møter med veileder, gruppen og oppdragsgiver. Dersom noen av oss ikke kan møte, skal det foreligge en saklig årsak og vedkommende plikter selv å informere gruppen så fort som mulig.
- Møte med veileder hver uke.
- Bli enige om hvem som skal føre referat fra møte før det begynner.
- Ved ukeslutt legges ny plan for neste uke, og en kort oppsummering fra uken som har vært.
- Signifikante avgjørelser skal tas sammen i gruppen.

- Være forbedredt og ha gjort arbeidsoppgavene sine. Informere gruppen om det har oppstått noen utforutsette problemer som gjør at man "henger etter"
- Konsekvenser for regelbrudd vil i første instans være en advarsel. Ved gjentatte regelbrudd vil saken tas opp med veileder.

Underskrift: Espen Myrum

Espen Myrum

Underskrift: Simen A. Nørstebø

Simen Nørstebø

C Prosjektavtale

NTNU

Vår dato Vår referanse

Norges teknisk-naturvitenskapelige universitet

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og Norsk Institutt for naturforskning (NINA)

_____ (oppdragsgiver), og

_____

_____ Simen Andre Nørstebø og Espen Myrum (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra _____01.01.2019_____ til_____16.05.2019_____ .

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne

prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): _____Sony George_____

Oppdragsgivers kontaktperson (navn): ____Jon Museth_____

Student(er) (signatur): Espen Myrvoll dato 29.01.2019

Sim A. Mikkelsen dato 29.01.2019

_____ dato 29.01.2019

_____ dato 29.01.2019

Oppdragsgiver (signatur): Jan Henrik dato 29.01.2019

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

Simen Andre Nørstebø, Espen Myrum

Automatisk gjenkjenning av settetisk

NTNU
Kunnskap for en bedre verden