Sjøkrigsskolen

Bacheloroppgave

Sonarens fremtid

– Auralklassifisering i variable miljøer ved hjelp av kunstige nevrale nettverk –

av

Mia Caroline Pavon Stavland

Levert som en del av kravet til graden:

BACHELOR I MILITÆRE STUDIER - FORDYPNING LEDELSE OG MARINE-INGENIØR VÅPEN, ELEKTRONIKK OG DATA

Antall ord: 15 875

Innlevert: desember 2023

Godkjent for offentlig publisering

Publiseringsavtale

En avtale om elektronisk publisering av bachelor/prosjektoppgave

Kadetten(ene) har opphavsrett til oppgaven, inkludert rettighetene til å publisere den.

Alle oppgaver som oppfyller kravene til publisering vil bli registrert og publisert i Bibsys Brage når kadetten(ene) har godkjent publisering.

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

Jeg (Vi) gir herved Sjøkrigsskolen rett til å gjøre denne oppgaven tilgjengelig elektronisk, gratis og uten kostnader	[x] Ja	[ ] Nei
Finnes det en avtale om forsinket eller kun intern publisering? (Utfyllende opplysninger må fylles ut)	[ ] Ja	[x] Nei
Hvis ja: kan oppgaven publiseres elektronisk når embargoperioden utløper?	[ ] Ja	[ ] Nei

Plagiaterklæring

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

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

Dato: 03 - 12 - 2023

Mia Caroline Pavon Starland

Kadett navn

Forord

Denne oppgaven er skrevet i perioden september til desember 2023 som et krav til graden i Militære studier med fordypning i ledelse og marineingeniør våpen, elektronikk og data.

Målet med oppgaven er å anvende relevant kunnskap innenfor kunstig intelligens og dataprosessering til å utvikle og teste et kunstig nevral nettverk på undervannsdata. For å gjøre dette benyttes fagstoff både i- og utenfor pensum.

Oppgavens overordnede tema er hvorvidt kunstig intelligens kan effektivisere auralklassifisering ved å trene et nevralt nettverk til å klassifisere skip i lydopptak fra hydrofoner. Om bord på ubåter er det i dag sonaroperatører som manuelt klassifiserer hvilke typer fartøy som befinner seg i farvannet rundt seg.

Arbeidet startet i september 2023 ved at relevante parter ble kontaktet for innhenting av informasjon om temaet. Blant annet ble Ubåtsenteret og forfatterne av *Sonaroperatøren* (2018), kontaktet. Ideen for temaet i oppgaven kom våren 2023 under idemyldring, men ble ikke fastsatt før i slutten av august 2023 etter diskusjoner med Aksel Wold Eide og Andre Adelsten Søvik. Ettersom temaet har blitt undersøkt tidligere ble det konkludert med at det kunne være interessant å undersøke flere av utfordringene som har blitt støtt på tidligere enda nærmere.

En takk rettes til mine veiledere på Sjøkrigsskolen, Alexander Sauter og Christophe Massacand. Takk til eX3 som tillot tilgang til maskinklyngen, og til Aksel Wilhelm Wold Eide i FFI for god veiledning, spesielt innenfor maskinlæring.

Mia Caroline Pavon Stavland

Mia Caroline Stavland

Sjøkrigsskolen 3. desember 2023

Oppgaveformulering

Hvor godt presterer et konvulsjonsnett, trent på data fra DeepShip, på klassifisering av fartøy i variable miljøer?

Sammendrag

Satsningen på kunstig intelligens (KI) predikeres at vil øke i årene framover. Det antydes i Forsvarskommisjonen, 2023, at KI sitt poensiale innenfor militær teknologi per i dag ikke utnyttes til det fulle. Denne oppgaven skal utforske anvendelsen av KI i en militær kontekst, mer konkret til auralklassifisering eller generell klassifisering av fartøyssignatur i hydroakustisk data.

Sonar (Sound Navigation and Ranging) er en sensor som benyttes til å kartlegge undervannsmiljøet sonaren befinner seg i ved å sende ut og motta signaler i form av lydbølger. Sonar er den viktigste sensoren om bord ubåter og benyttes til navigering og overvåking av andre fartøyer. I dag klassifiseres fartøyer manuelt av en sonaroperatør.

Oppgaven undersøker utfordringene ved bruk av KI til klassifisering av fartøyssignatur ved å teste hvor godt et kunstig nevralt nettverk presterer på data som varierer i geografisk lokasjon og årstid. Hydroakustikk er et komplekst fagfelt med variabler som endrer seg konstant. Dette, i kombinasjon med lite tilgang på relevant og mangfoldig data, gjør det vanskeligere å utvikle og implementere systemer som er pålitelige nok i en militær kontekst.

KI-algoritmer krever stor datakapasitet, noe som ikke eksisterer på Sjøkrigsskolen i dag. En ekstern maskinklynge er et alternativ for å gjennomføre forsøk uten å måtte anskaffe kraftige datamskiner som blir utdatert etter kort tid.

Oppgaven trener to ResNet-18-nettverk på data innhentet av en statisk hydrofon ved hjelp av en ekstern maskinklynge. Deretter testes nettverkene på sine individuelle testsett, det ene bestående av data fra en annen geografisk lokasjon og den andre fra en annen årstid.

Forsøkenes resultater kan brukes til å argumentere for at mangelen på store og representative datasett et stor hinder for å utvikle et nevralt nettverk som er pålitelig også når geografisk lokasjon og årstid endrer seg. Kunstige nevrale nettverk krever store mengder data for å bli trent til utføre komplekse oppgaver. I tillegg kreves det at dataen er behandlet tilstekkelig for å lage bedre forutsetninger for at mønstrene i dataen gjenkjennes.

Innholdsfortegnelse

Figurer

Tabeller

Forkortelser

- **SONAR** Sound Navigation and Ranging
- **ML** Machine Learning *norsk maskinlæring*
- **KI** Kunstig intelligens
- **AI** Artificial Intelligence
- **CNN** Convolutional neural network
- **ResNet** Residual network
- **RU** Residual Layer
- **Slurm** Simple Linux Utility for Resource Management
- **HPC** High Performance Computing
- **STFT** Short Time Fourier Transform
- **FFT** Fast fourier transform
- **FP** falsk positiv
- **FN** falsk negativ
- **TP** sann positiv *eng. true positive*
- **TN** sann negativ *eng. true negative*

Innledning

I Forsvarskommisjonen av 2021 sin utredning, lagt fram for regjeringen i mai 2023, står det:

> *«Kunstig intelligens vil bli det viktigste utviklingsfeltet de neste to tiårene.»*
> (Forsvarskommisjonen, 2023, s. 122)

Kunstig intelligens (KI) og maskinlæring (ML) kan i fremtiden vise seg å være en svært relevant teknologisk utvkiling innenfor sensorteknologi. En av grunnene til dette kan være at KI og ML kan bidra til et bedre grunnlag for autonome systemers beslutningsprosesser. Med kunstig intelligens menes datasystemer som lærer og dermed fremstår som intelligent. Det å utvikle teknologi innenfor KI kan være viktig for at Norge og Norges allierte skal være rustet til å møte fremtidens teknologi. I Forsvarskommisjonens utredning nevnes ordet *«kunstig intelligens»* over femti ganger, som i tillegg til anbefalingene tilsier at fagfeltet er et fokusområde både i Norge og hos våre allierte (Forsvarskommisjonen, 2023, s. 203). I Forsvarssjefens fagmilitære råd pekes det på at den teknologiske utviklingen går veldig raskt, noe som stiller krav til oppdateringer og videreutvikling av våre militære systemer dersom de skal ha relevans (Forsvaret, 2023, s. 24).

Norge er i dag utstyrt med ubåter fra ULA-klassen som opererer i norske farvann og bidrar til det strategiske målet om avskrekking. Ubåtenes viktigste sensor er sonar, som brukes til navigasjon og til identifisering av hva som befinner seg i ubåtens omgivelser. I følge Forsvarskommisjonen, 2023, vil

> *under vann ... trolig være et av få steder det fortsatt vil være mulig å operere i skjul. Det er en av hovedgrunnene til at de mektigste aktørene i verden fortsetter å satse tungt på undervannsteknologi og undervannsbåter.*

Undervannsdomenet vil trolig bli enda viktigere enn det allerede er, som gjør at teknologiutviklingen må tilpasses deretter. Både økt sensordekning og ubåters evne til å behandle dataen som samles inn vil være et viktig steg mot å være i bedre stand til å ha kontroll på kystområdene under vann. Forsvarskommisjonen beskriver godt hvorfor utvikling innenfor KI kan forbedre vår evne til overvåke Norges kyst:

> *Det blir mulig å forbedre overvåking over og under vann. Overvåking av undervannsaktiviteter er spesielt utfordrende, og Norge har gode forutsetninger for å utvikle nye systemer. Nye sensorsystemer, kombinert med kunstig intelligens og stordata-analyse, vil kunne oppdage trusler tidligere og over større områder. Slik kan vi håndtere truslene effektivt, og vi reduserer utfordringen med et lite forsvar, store havområder og en langstrakt kystsone å overvåke og beskytte.*(Forsvarskommisjonen, 2023, s. 190).

Norge omtales som «NATO i nord» fordi vi ligger lengst nord i alliansen og har store deler av vårt operasjonsområde i Arktis. Vår nabo i øst, Russland, har også store deler av sin maritime aktivitet i nord, blant annet ved at de har posisjonert sine strategiske atomubåter på Kolahalvøya, kun ca. hundre kilometer fra norskegrensen. Som følge av den geografiske lokasjonen til Russlands base i nord kan det forutsees at Russland har som mål å kunne operere mest mulig skjult både lokalt og i sine ubåters transitt ifra nordlige farvann til Atlanterhavet. Det nevnes på side 146 i Forsvarskommisjonen, 2023, at moderne ubåter er stillegående og ressurskrevende å følge. I tillegg konstanterer Forsvaret, 2023, at i møte med russlands stadig mer moderne, stillegående og slagkraftige ubåter vil

> *...evne til anti ubåtkrigføring, i samvirke med allierte, spesielt USA og Storbritannia, ... derfor [være] en prioritert kapabilitet også i fremtiden* (Forsvaret, 2023, s. 62).

For å være i stand til å følge opp målsetningen om å være i stand til å overvåke ubåtaktivitet i nord er det å satse på teknologi, deriblant KI, et anbefalt tiltak.

Sonar er en viktig sensor for å overvåke omgivelsene under vann fordi den kan sende ut og oppfatte lydbølger i vann for å kartlegge undervannsmiljøet. I dag er sonarer utstyrt med verktøy innenfor signalbehandling som gjør manuell analyse mer effektiv og nøyaktig, men klassifiseringen skjer av mennesker ved å lytte til lydsignalene,

såkalt auralklassifisering. I dag trenger sonaroperatører opplæring og lang erfaring for å gi gode prediksjoner på fartøy. Spørsmålet om hvorvidt KI kan supplere til, eller potensielt erstatte, dagens form for skipsklassifikasjon er interessant fordi et pålitelig KI-system vil potensielt være i stand til å klassifisere skip raskere enn mennesker. I tillegg kan et slikt verktøy i kombinasjon med autonome systemer lede til utviklingen av nye ubemannede sensorplattformer. Autonomi er er et fagfelt innenfor informatikk som innbefatter systemer som designes for å helt eller delvis operere selvstendig uten menneskelig interaksjon.

Det er ikke kun ubåtkrigføring som er relevant i forbindelse med implementeringen av et KI-system som analyserer undervannsdata. Norge har betydelige økonomiske interesser og svært verdifull infrastruktur både over og under vann langs hele kysten. Dette skal, i tillegg til annen aktivitet langs norskekysten, overvåkes og forsvares. Med store havinstallasjoner i Nordsjøen og Norskehavet er det ikke utenkelig at et slikt system kan bistå i overvåkning av disse anleggene, i tillegg til å monteres ved statiske ukjente lokasjoner både for å samle inn data og å drive overvåkning. I tillegg satser både NATO, USA, Russland og Kina stort på KI, som understreker hvor betydningsfull teknologien kan bli i fremtiden (Ulrich Jochheim, 2021).

Til tross for stor utvikling og stor satsning innenfor KI-fagfeltet er det flere hinder som står i veien for at KI kan bli implementert for å effektivisere auralklassifisering om bord militære fartøyer i dag. I DataFlair, 2021, forklares ulempene i maskinlæring og nevner relavante punkter om hvorfor den menneskelige sonaroperatøren ikke har blitt erstattet av en datamaskin. Maskinlæring er et underfelt i KI som innbærer at datamaskiner finner mønstre i store datamengder og er belyst i kapittel 2.2. Videre listes utfordringene med en forklaring.

Godt datagrunnlag

ML-algoritmer trenger store mengder data som er relevant, og av høy kvalitet uten bias (DataFlair, 2021). Dette eksisterer ikke innenfor militær kontekst i åpne forum, og blant annet er data av skipssignatur i undervannsdata lite tilgjengelig. Gradering er et hindrer for at data kan bli delt på tvers av institusjoner til, for eksempel, å lage ett stort og mangfoldig datasett. Sikkerhetsrisikoen er stor, som gjør at de fleste nasjoner holder kortene tett om brystet om hvilken teknologi de innehar og generer sin egen data i liten

skala. Den første utfordringen er at det finnes for lite og for mangelfull data. I tillegg er undervannsakustikk kjennetegnet av å være et komplekst fagfelt med mange variabler som endrer seg konstant. Dette gjør det desto viktigere at datasettene er mangfoldige og relevante.

Tid og ressurser

Det å lage og utvikle en ML-algoritme som fungerer optimalt er veldig tidkrevende og krever mye kompetanse og ressurser (DataFlair, 2021). I tillegg krever det mye maskinkraft å utvikle systemer som er trent på så store mengder data som trengs for å lage et system som oppfyller kravene til et militært system.

Robusthet mot feil

ML-algoritmer er veldig sensitive og må trenes opp spesifikt og for oppgaven den skal løse. I en militær kontekst har man ikke råd til feil, og datasystemene må oppfylle et høyt krav til robusthet og troverdighet. En ubåt er i bevegelse store deler av tiden og møter på store variasjoner i miljøet den befinner seg i. Det å oppholde seg i en fjord sammenlignet med åpent hav kan, uten tilstrekkelig data, føre til usikkerhet i systemets prediksjoner hvis den ikke er trent godt nok.

Oppsummert belyser de ovennevnte utfordringene hva som følger med det å utvikle pålitelige ML-systemer uansett hvilken oppgave det skal løse. Denne oppgaven søker å undersøke det som trolig er de største årsakene til at forskningen på anvendelsen av ML til auralklassifisering fremdeles er begrenset.

Oppgaven tar utgangspunkt i tidligere forskning på området, spesielt forskningen gjort av Irfan et al., 2021. Arbeidet brukes som et utgangspunkt for metodene brukt i denne oppgaven, og er kilden til primærdatasettet DeepShip. I tillegg har lignende forskning blitt gjort av Gimse, 2017 som testet ulike maskinlæringsalgoritmer for å finne den beste egnede til å klassifisere skip. Et tredje eksempel er et prosjekt gjort på Sjøkrigsskolen av Heimli og Vegusdal, 2018. Fellesnevneren for de to sistnevnte er mangelen på relevant data.

I tillegg til de tre ovennevnte er det flere artikler som beskriver lignende forskning som benytter bildefremstilling av lyd som input til algoritmen. Det er vanskelig å finne

omfattende forskning på dette i en militær kontekst, dog kan det antas at dette er på grunn av graderingsnivå eller eventuelt på grunn av mangelen på slik forskning.

1.0.1 Problemstilling

For å undersøke i hvor stor grad utfordringene påvirker anvendelsen av nevrale nettverk vil oppgaven undersøke følgende problemstilling:

> ***Hvor godt presterer et konvulsjonsnett, trent på data fra DeepShip, på klassifisering av fartøy i variable miljøer?***

Ved å utvikle, trene og teste et ML-system er hensikten å identifisere hvor godt eller dårlig systemet er til å tilpasse seg faktorene som kan påvirke systemets nøyaktighet. Systemet skal testes to ganger på to datasett som er generert i enten ulike geografiske lokasjoner eller ulike årstider.

Variable miljøer kan bety mangt, men er i denne oppgaven definert av to parametere som vil være utgangspunktet for forskningen. Parameterne er geografisk lokasjon og årstid.

1.0.2 Målsetning

Målet for oppgaven er å ta utgangspunkt i arbeidet og metoden til Irfan et al., 2021, for å lage og teste er kunstig nevral nettverk anvendt til å klassifisere fartøy i undervannsopptak. Irfan et al., 2021 har forsket på bruken av KNN til å klassifisere fartøy i undervannsopptak fra en stasjonær hydrofon.

Siden problemstillingen går ut på å teste hvor godt et nevralt nettverk kan anvendes i variable undervannsmiljøer stiller ikke dette krav til at datasettet må bestå av militære data. I stedet vil data fra åpne kilder fra forskjellige statiske hydrofoner benyttes til å simulere et endret undervannsmiljø. Datasettene som benyttes vil bli introdusert i kapittel 3.3.1 og 3.3.2.

Målsetningen er å gjennomføre to separate forsøk, forsøk 1 og forsøk 2 som begge undersøker hva endringer mellom trening- og testdatasettet har å si for nettverkets prestasjon. Prestasjon er knyttet til målene *accuracy*, *Precision*, *recall* og *F1-score*.

Forsøk 1 skal trene et nettverk på data fra en lokasjon og deretter testes med data fra en helt annen lokasjon. Hensikten med forsøket er å teste hvor godt et KNN presterer

når den testes på data fra et annet geografisk miljø. Forsøk 2 vil også bestå av å trene et KNN, men trenes på data innhentet i vinterhalvåret og testes deretter på data innhentet fra sommerhalvåret. Målet er å teste om det KNN kan anvendes på data fra en annen årstid enn den er trent til.

1.0.3 Begrensninger

Oppgaven vil benytte seg av kun en type nevralt nettverk og vil bruke fine-tuning. I tillegg er oppgaven begrenset til å bruke en form for visuell fremstilling av lyddata i preprosesseringen og tar ikke hensyn til avstandsdata annet enn at opptakene er tatt innenfor en 2km radius. Metoden er begrenset til kun bildegjenkjenning og visuelle lyddata for å klassifisere fartøy i lydopptak under vann.

Teori

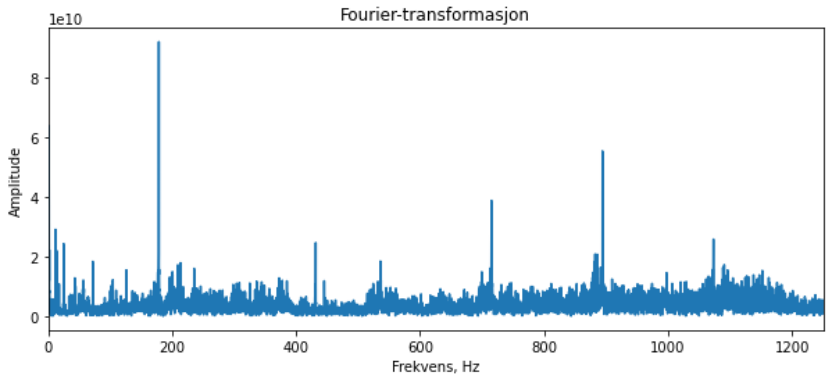
2.1 Sonar

Sonar er en sensor som brukes i flere fagfelt, også utenfor maritime operasjoner. En sonar både sender og mottar lydbølger under vann for å klassifisere og lokalisere objekter (Waite, 2002, s.xvii). En sonar består av en transduser, som genererer lydbølger, og et sett med hydrofoner, som oppfatter lydbølger. Man skiller mellom aktiv og passiv sonar, der aktiv sonar kjennetegnes av at sonaren har en utsending som hydrofonene mottar ekkoet fra. Når de utsendte lydbølgene reflekteres på, for eksempel et annet skrog, vil refleksjonen mottas av sonarens hydrofoner. Ved å bruke lydens hastighet og tiden det tar for utsendingen å sendes, reflekteres og mottas kan objektets avstand regnes ut. Ulempen med aktiv sonar er risikoen for deteksjon ved at utsendelsen i verste fall mottas av en fiende. Passiv sonar kjennetegnes av at det ikke benyttes aktiv utsendelse, men at det lyttes etter lydbølger fra omgivelsene. På denne måten kan sonar brukes til å overvåke lyder i området uten å avgi signatur.

2.1.1 Lydbølger

Det er relevant å gå gjennom overordnet teori bak lydbølger for å kunne forklare teorien bak spektrogram. Lyd består av langsgående trykkbølger som forplanter seg gjennom alle materialer. Lyd har to egenskaper, frekvens og amplitude, der førstnevnte forteller hvor mange svingninger bølgen har i løpet av ett sekund, og sistnevnte hvor høy energi svingningene har (intensitet).

I de aller fleste tilfeller vil ikke lyd bestå av kun én frekvens, men av mange frekvenser som summert omfatter lydens frekvensspekter. Ved å vite hvilke frekvenser et lydsignal



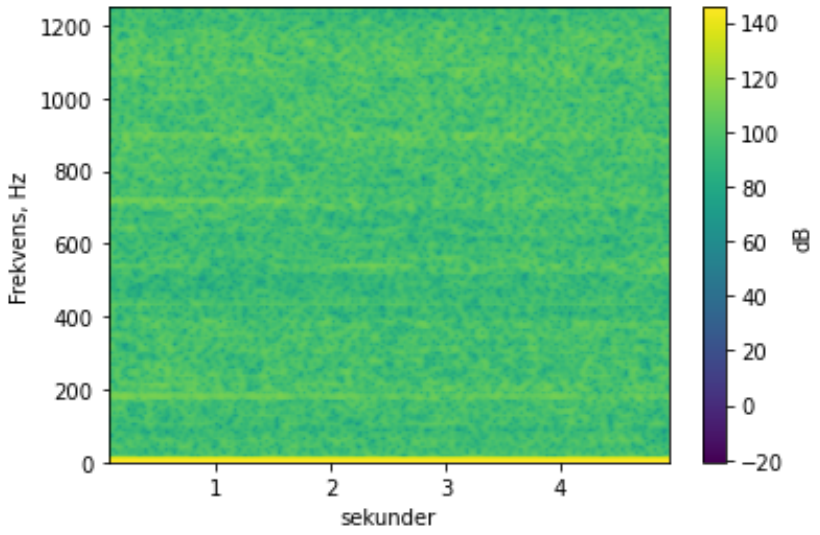
Figur 2.1.1: Plot av amplitude over frekvens i en vilkårlig skipspassering plottet ved hjelp av Fast Fourier Transform.

består av er det mulig å skille ut frekvensene som er av interesse, som i vårt tilfelle er de som kjennetegner en skipsmotor. Det er mulig å regne ut hvilke frekvenser et lydsignal består av og sette det sammen ved å utføre en fouriertransformasjon av signalet. Figur 2.1.1 vises en vanlig Fast Fourier Transform (FFT) som trekker ut alle frekvensene mellom 0-1,250kHz i signalet. Ulempen med FFT er at den ikke viser variasjon i frekvens over tid. For å se hvordan frekvensene endres i løpet av lydklippet benyttes short-time fourier transform (STFT).

STFT

Short-time fourier transform gjør det mulig å analysere frekvensvariasjoner over tid ved at den anvender FFT på korte tidsintervaller i stedet for hele lydklippet (Collimator, 2023). Dette produserer et spektrogram som har tre akser: frekvens, tid og amplitude. Tidsintervallene som brukes kan være små nok til at spektrogrammet viser endringene som om de var kontinuerlig. Figur 2.1.2 viser et eksempel på et spektrogram, og har aksene Hz, s og dB.

Variabelen n_{fft} kan endres for å bestemme hvor korte tidsintervaller STFT skal utføres på, omtalt som hvor store eller små *vinduer* som benyttes. Når den økes, øker også oppløsningen i spektrogrammet. For å definere hvilke datapunkter som inngår i hvert vindu sendes dataen gjennom en window function (Gupta, 2017). Hanning-funksjonen er en type window function som kan benyttes med FFT.



Figur 2.1.2: Frekvens over tid plottet ved hjelp av STFT med hanning-funksjon, også omtalt som et spektrogram.

Lydnivå

Amplitude oppnevnes i decibel (dB) og er en skala som oppgir lyd logaritmisk relativt til hva menneskeøret er i stand til å høre. 0 dB betyr at lyden er så lav at den ligger på terskelen for hva mennesket kan høre. Formelen for konverteringen fra W/m til dB er gitt i formel 2.1. I står for intensiteten målt i lydsignalet, I_0 er referanseverdien for 0 dB, som er $10^{-12}W/m$.

$$L(dB) = 10\log\left(\frac{I}{I_0}\right) \tag{2.1}$$

Lydintensitet avtar proporsjonalt med avstanden R. Forholdet er gitt i formel 2.2. Dersom avstanden dobler betyr det at lydintensiteten synker til en fjerdedel (Petter Brækken, 2005, s. 10).

$$I = \frac{P}{A} = \frac{P}{4\pi R^2} \tag{2.2}$$

2.1.2 Hydroakustikk

Lydbølgers fart gjennom vann er ikke konstant, men varierer ut ifra kombinasjonen av vannets saltholdighet, temperatur og trykk (dybde) (Waite, 2002, s. 49). Denne variasjonen kan i stor grad påvirke hvordan lyden oppfattes av sensorer og mennesker.

En annen faktor som påvirker oppfattelsen av lydbølger er undervannstopologien (havbunnen) som forårsaker etterklang (reverberasjon). Dette er fordi lyden reflekteres gjentatte ganger, noe som kan forstyrre signalet (Waite, 2002, s. 41). Undervannstopologien langs Norskekysten kjennetegnes av mye fjell, fjorder, grunner og ujevnheter, noe som fører til mer etterklang enn i åpent hav.

En tredje faktor som påvirker oppfattelsen av signaler er at sonarer opererer i omgivelser med mye støy(Waite, 2002, s. 90). De største støykildene er:

- Sonarens egenstøy
- Ambient noise – konstant støy i havet
- Egenstøy fra fartøyet sonaren er montert på
- Støy fra biologisk aktivitet og menneskelige konstruksjoner

Resultatet av at det er stor variasjon i undervannsmiljøet basert på mange faktorer gjør at det er vanskeligere å gjenkjenne mønstre i en bakgrunn som alltid endrer seg. Dette er en viktig faktor for implementeringen av nevrale nettverk til å gjenkjenne skipstyper i l undervannsdata fordi det er vanskelig å finne data som er dekkende for alle miljøer i alle årstider.

2.1.3 Signatur og kavitasjon

Signatur innenfor undervannsakustikk er begrepet for et fartøys utsendelse i form av lyddølger, Begrepet omfatter lyden fartøyet sender ut gjennom vannet og hvilke frekvenser den består av. Fartøyets lydsignatur består av propell-støy, bølgebrytning og intern maskineri (Mark, 2021, s. 3).

Kavitasjon er den mest framtredende kilden til støy fra et fartøy og bidrar til å gjøre den gjenkjennbar i lydbildet ved at alle skip har en unik signatur (Gimse, 2017, s. 6). De aller fleste fartøy i verden drives fremover av en eller flere propeller som omformer et dreiemoment til skyvekraft (T. Editors of Encyclopaedia, 2023) Motorens rotasjon i kombinasjon med propellens form fører til at vannet dyttes mot en side av propellbladet, imens det på andre siden formes et negativt trykk. Det lave trykket gjør at gass som ligger i lavtrykksområdet former bobler. Det er når gassboblene kollapser at det avgis

en lyd, som i store volum produserer det som blir en særegen signatur for den spesifikke motoren og propellen (Gimse, 2017).

Fartøyssignatur varierer, men typisk er det de lavere frekvensene som er mest fremtredende, nemlig mellom 200-500Hz (Mark, 2021). I denne oppgaven omtales frekvensene som har høyest amplitude i signalet som *hovedfrekvenser.* Disse frekvensene er synlige i spektrogrammene som vannrette linjer langs tidsaksen.

2.1.4 Digitalisering

Lydbølger en mekanisk energioverføring i et materiale, i dette tilfellet i vann. For å måle lyd under vann brukes en hydrofon som detekterer svingninger og overfører energien til spenning. For å gjøre opptak av lyd og lagre det digitalt må lyden overføres fra en analogkontinuerlig fremstilling til digital/diskret. For å gjøre dette tas det diskrete målinger med en viss rate som heter samplingsfrekvens. Dette er relevant i forhold til hvilken samplingsfrekvens, f_s (kHz) hydrofonen har, som kan variere ut ifra hvilken spesifikk modell hydrofonen er og dens tekniske spesifikasjoner.

2.1.5 Auralklassifisering

Begrepet auralklassifisering[1] beskriver konseptet å klassifisere lydkilder ved bruk av hørselen og eventuelle signalbehandlingsverktøy. Dette er en svært viktig del av ubåtkrigføring fordi det er en avskrekkende faktor at en ubåt kan detektere og overvåke andre fartøyer uten å bli oppdaget. Som nevnt tidligere i kapittelet kan fartøyer gjenkjennes av frekvensen til motoren/kavitasjon. Det er gjennom opparbeidet erfaring og trening at en sonaroperatør effektivt kan klassifisere skipstyper og identifisere fartøy innenfor sonarens rekkevidde ved å lytte til fartøyenes signaturer. Denne prosessen er ikke uten usikkerhet, som er grunnlaget for at et ML-verktøy kan skape større pålitelighet i klassifiseringen og senke tiden det tar for en operatør å klassifisere fartøyer.

2.2 Maskinlæring

Maskinlæring (ML) er et stort fagfelt innenfor informatikk som omhandler datamaskiner som "lærer". I denne oppgaven blir en ML-algoritme brukt til klassifisering av

[1] *"Aural"* defineres på engelsk som: "relating to the ear or the sense of hearing.(“Aural”, n.d.)

bilder, som innebærer at den skal kategorisere dataen og fordele dem i et forhåndsbestemt antall diskret kategorier. Delkapittelet belyser og forklarer de mest grunnleggende og relevante områdene innenfor ML.

2.2.1 Grunnleggende om maskinlæring

Maskinlæring er en underkategori av kunstig intelligens (KI), som innebærer at datamaskiner lærer ved å trenes på å gjenkjenne mønstre i datasett (Tidemann & Elster, 2023). En overordnet definisjon av Arthur Samuel kom allerede på femtitallet:

> *[Machine learning is] the field of study that gives computers the ability to learn without explicitly being programmed* (Brown, 2021).

Selv om KI og ML som fagfelt ble introdusert allerede på 50-tallet, har bruken ikke økt før 2000-tallet grunnet den senere utviklingen av kraftig datakapasitet og fremveksten av store digitale datasett. Det er denne utviklingen som har gjort at ML har blitt bruk i økende grad de siste tyve årene (Tidemann, 2023). I tillegg har bruk av GPU ført til at maskiner kan trenes enda mer effektivt.

Det finnes mange typer maskinlæringssystemer som kategoriseres etter måten de lærer på. Denne oppgaven vil ta for seg modellbaserte og veiledede maskinlæringssystemer, deriblant konvolusjonsnettverk *eng. Convolutional Neural Networks*.

2.2.2 Veiledede og modellbaserte ML-systemer

Veiledede ML-systemer (Supervised learning) lærer ved å bruke data som er annotert med fasit. Fasiten gjør det mulig for algoritmen å måle sin egen prestasjon og følgelig kunne optimalisere metoden sin ved å bruke fasiten som respons. Systemet må trenes med store mengder annotert data for å være i stand til å gi nøyaktige prediksjoner (Géron, 2023, s. 10).

Modellbaserte systemer (Model-based) bruker treningsdataen til å generalisere "problemet" ved å lage en matematisk modell. Denne modellen brukes så til å lage gode prediksjoner for usett data. Et enkelt eksempel på dette er å bruke lineær regresjon til å predikere verdien på nye datapunkter gitt at treningsdataen fornuftig kan modelleres av en enkel matematisk lineær funksjon. Nøkkelen til å lage gode maskinlæringssystemer

ligger i generalisering, altså hvor godt den utvinnede matematiske modellen representerer virkeligheten og kan anvendes på andre eksempler enn det den har blitt trent på.

2.3 Trening

Trening er prosessen for hvordan en datamaskin *lærer*. En måte å definere dette på er:

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.* (Goodfellow, Bengio, & Courville, 2016, s. 97)

Kort forklart jobber maskinen seg fram til den beste løsningen på en oppgave ved å få tilbakemelding på hvor godt eller dårlig den presterer underveis i treningen. Forskjellen mellom ML og en programmert datamaskin er at den førstnevnte lærer seg hvordan en oppgave skal utføres, imens en programmert datamaskin utfører oppgaven etter hvordan den er forhåndsprogrammert av mennesker. I de neste delkapitlene blir grunnleggende konsepter innenfor ML belyst og forklart overordnet.

2.3.1 Kostfunksjonen

Kostfunksjonen *eng. Cost Function* regner ut hvor god modellen er på å predikere riktig. Som tidligere nevnt er ML-systemer designet til å lære ved å få tilbakemelding på hvor godt den presterer *eng. performance measure*. Kostfunksjonen er en matematisk funksjon som kvantifiserer målet på prestasjon ved, for eksempel, å måle differansen mellom modellens predikerte output og dataeksemplenes reelle output (fasiten). En modell med høy verdi på kostfunksjonen presterer dårligere enn en modell med lav verdi. Følgelig betyr dette at det å minimere kostfunksjonen resulterer i høyere *performance*, tilsvarende at modellen representerer treningsdataen i større grad. Selve formelen for kostfunksjonen varierer ut ifra hva algoritmens tenkte bruksområde er, men inneholder data-input og modellens parametere.

2.3.2 Gradient descent

Gradient descent er sentral i ML fordi det er prosessen som optimaliserer verdiene på parameterne slik at kostfunksjonens output blir minst mulig. Kostfunksjonen gir en respons på modellens prestasjon og brukes til å justere modellen slik at den ved neste forsøk presterer bedre. Variablene som justeres heter parametere og representeres som en vektor $\vec{W}_{ij}$.

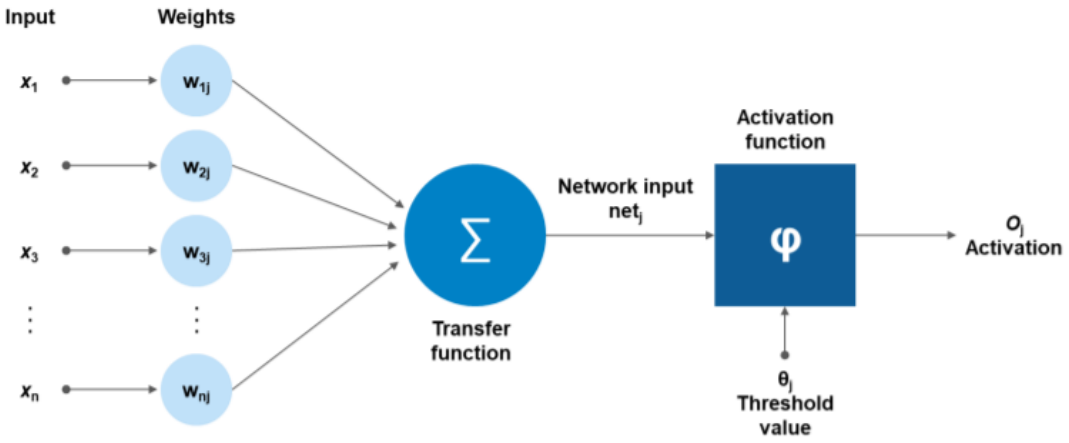
2.3.3 Overfitting og underfitting

Overfitting og *underfitting* er begge uønskede tilfeller av at modellen ikke generaliserer godt fordi den enten er overtilpasset treningsdataen, overfit, eller at den er undertilpasset, underfit. Et eksempel på overfitting er når algoritmen plukker opp støy i dataen som en del av generaliseringen, noe som vil gjøre at modellen blir tilpasset treningsdataen fordi den memorerer både data og støy. Selv om modellen er godt trent på treningsdataen vil modellen prestere dårligere når den anvendes på ny data uten tilsvarende støy. Overfitting skjer når modellen er altfor kompleks i forhold til størrelsen på datasettet og støy i dataen (Géron, 2023, s. 31). Det motsatte av overfitting er når modellen er for enkel til å gjenkjenne avanserte mønstre i dataen og generaliserer dårlig som følge av dette. For å få en modell som generaliserer dataen på en god måte kreves et system som er godt tilpasset behovet og at treningsdatasettet er stort nok, uten støy og representativt (Géron, 2023, s. 33).

Et tiltak for å redusere overfitting er regularisering. Når det er veldig mange parametere i en modell kan den blir for kompleks i forhold til treningsdataen. Regularisering begrenser effekten støy i dataen har i modellens trening for å unngå at den trenes til å gjenkjenne støy (Géron, 2023, s. 392).

2.3.4 Nevrale nettverk og Deep Learning

Kunstige nevrale nettverk (KNN) er inspirert av hjernens arkitektur og er ML-algoritmer bestående av kunstige nevroner koblet sammen i et nettverk. KNN er noen av de mest brukte ML-algoritmene i dag og kjennetegnes av å være allsidige, kraftige og skalerbare (Géron, 2023, s. 299). Nevrale nettverk er komplekse og krever store mengder data og maskinkraft for å trenes godt og innenfor rimelig tid (Géron, 2023, s. 301).

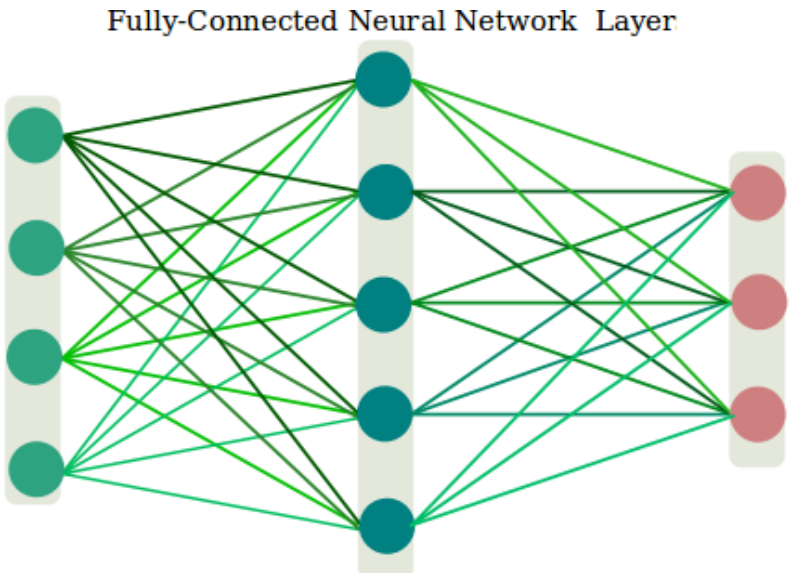


Figur 2.3.1: Diagram av en kunstig nevrons struktur. $\vec{x}$ er nevronens input, $\vec{w}$ inneholder parameterne, φ er output fra aktiveringsfunksjonen og θ_j er terskelverdi. Parameterne vektes i transferdunksjonen og ouputen sendes så gjennom activation function som gir nevronens output (Buettgenbach, 2021).

Det kunstige nevronet har en eller flere binære input og en binær output. En nevrons output blir aktiv dersom inputen til aktiveringsfunksjonen gir en verdi over terskelverdien θ_j. I diagram 2.3.1 vises hvordan et kunstig nevron er bygget opp. På venstre side vises de mange inputene $\vec{x}$ som er nettverkets input og består av alle datapunktene i treningssettet. I denne oppgaven består hver verdi i $\vec{x}$ av et bilde på 224×224 piksler. $\vec{W_j}$ inneholder alle parameterne for det respektive laget i nettverket. I transferfunksjonen vektes input-verdiene i $\vec{x_j}$ ut ifra verdiene på parameterne i $\vec{W_j}$ og summen sendes så gjennom aktiveringsfunksjonen som gir output basert på om verdien er høyere eller lavere enn terskelverdien θ_j. Det går an å velge mellom mange ulike aktiveringsfunksjoner. Den mest relevante for denne oppgaven forklares i kapittel 2.3.6.

Det kunstige nevronet er byggeklossen i kunstige nevrale nettverk som består av flere lag av kunstige nevroner, de fleste av dem "skjult" (hidden layer). I et lag som er *Fully Connected* (også: *dense layer*) vil alle nevronene i nåværende lag få input fra alle nevroner i forrige lag og gi output til alle nevroner i neste lag. Slike lag finnes som regel i slutten av konvolusjonsnett (Rastogi, 2023). Figur 2.3.2 er en forenklet illustrasjon av en fully connected lag i et nevral nettverk.

Et KNN bestående av mange skjulte lag omtales som dype nevrale nettverk (Géron, 2023, s. 309). Konvolusjonsnett er en type dype nevrale nettverk som benyttes til bildeklassifisering og er spesialdesignet for å få bilder som input. Konvolusjonslag analyserer kun en liten del av hele bildet om gangen og gjenkjenner på denne måten



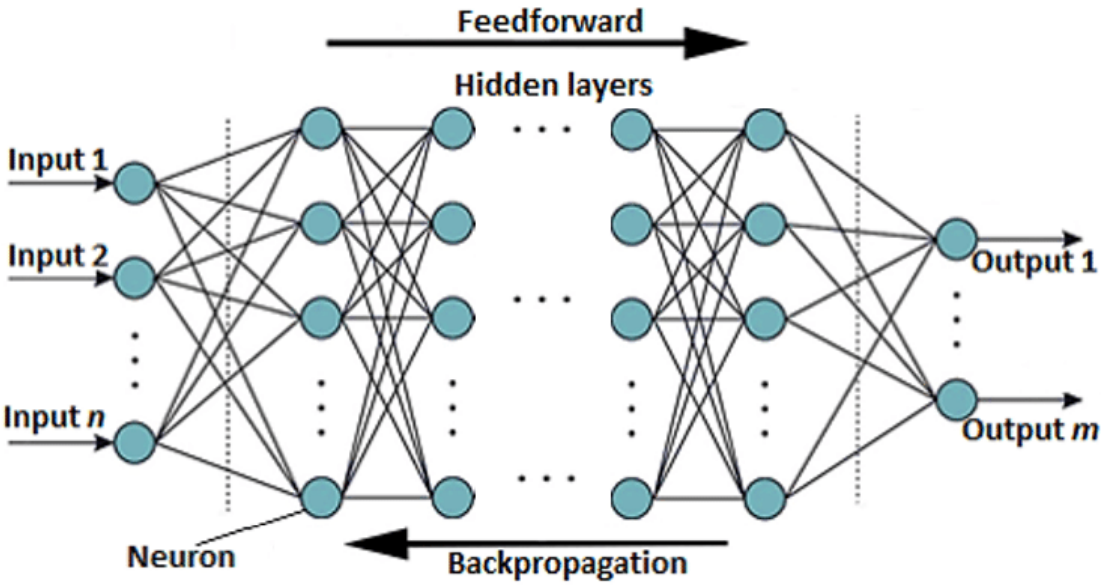
Figur 2.3.2: Forenklet illustrasjon av et fully connected lag. Illustrasjonen viser et lag av navroner som får bruker input fra alle nevroner i forrige og neste lag.

mindre trekk i bildet.

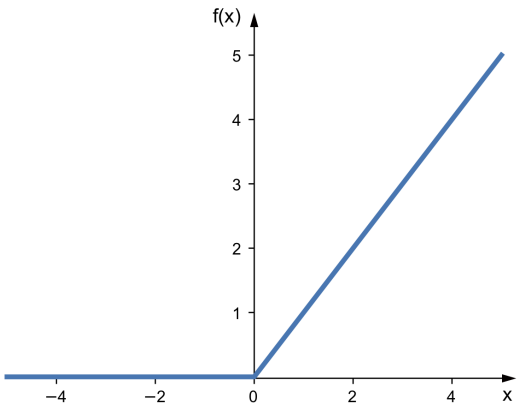
Dype nevrale nettverk benytter *backpropagation* for å justere parameterne i underveis i treningen. Algoritmen regner ut gradient ved å gå gjennom hele treningsdatasettet i bolker kalt *mini-batches*, et antall bestemt på forhånd. Hver gjennomgang av hele datasettet heter en *epoch*. *Forward propagation* skjer først, som innebærer at output regnes ut og lagres i alle lag, fra første til siste, altså fremover. Når siste output regnes ut blir kostfunksjonen regnet ut og basert på denne går algoritmen gjennom nettverket, fra siste til første lag, for å regne ut hvor mye hver parameter må justeres for å minke feilen. Til slutt gjennomføres selve justeringen som ble utregnet gjennom *backward propagation* (Géron, 2023, s. 311). Figur 2.3.3 viser en generell struktur på et nevralt nettverk med n *fully connected* lag og et input - og output-lag. I hver nevron regnes parameterne ut på nytt for hver mini-batch. Størrelsen på mini-batch er en av flere hyperparametere som kan justeres for å tilpasse nettverket til eget behov. Andre eksempler på hyperparametere er *learning rate* som påvirker gradient descent, og regulariseringsparameter som regulariserer nettverket.

2.3.5 ReLU

ReLU står for Rectified Linear Unit og er en aktiveringsfunksjon som er veldig mye brukt i KNN. Det er en delvis lineær funksjon som er vanlig å bruke til gradient descent



Figur 2.3.3: Strukturen til et nevralt nettverk med nevroner, skjulte lag, input- og output-lag (Abueidda, Lu, Qiyue, & Koric, Seid, 2020).

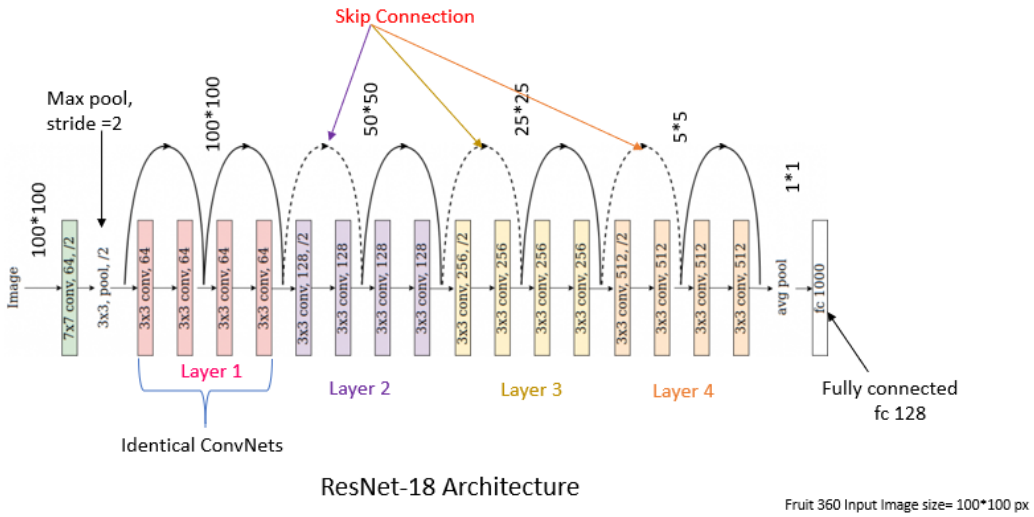


Figur 2.3.4: ReLU-funksjonen plottet mellom $(-5, 5)$. (Raschka, 2023)

med backpropagation. ReLU-funksjonen er relativt enkel og returnerer 0 dersom input-verdien er lavere enn 0. Dersom input-verdien > 0 er ouput-verdien lik input. For input > 0 oppfører funksjonen seg som en lineær funksjon, men er ikke-deriverbar i punktet $(0, 0)$ som vist i figur 2.3.4 (Jason Brownlee, 2020) . ReLU-funksjonen skrives enkelt på måten: $ReLU(z) = max(0, z)$ (Géron, 2023, s. 312).

2.3.6 ResNet

Nettverket benyttet i denne oppgaven heter Residual Network (ResNet). Kapittelet vil ikke gå i dybden på hvordan dette nettverket er bygget opp men tar for seg det mest sentrale.



Figur 2.3.5: Struktur av ResNe-18. Det siste laget er et FC-lag (Gaurav Singhal, 2020).

Nettverket ble kåret til vinner av ILSVRC[2] i 2015 (Géron, 2023, s. 505) med ResNet-152, som betyr et ResNet med 152 lag. Nettverket kjennetegnes av å ha såkalte *skip connections* som bidrar til at nettverket fungerer godt også når det blir dypere. ResNet består av mange *residual units* (RU) som er bygget opp av to konvolusjonslag hver. Nettverket bruker ReLU-funksjonen som aktiveringsfunksjon. Figur 2.3.5 viser strukturen til et ResNet-18-nettverk og viser *skip connections* mellom hvert par av konvolusjonslag i en RU. Det siste laget er et *fully connected*-lag og kan trenes separat fra resten av nettverket. Dette forklares mer i kapittel 2.3.7.

2.3.7 Fine-tuning

Fine-tuning er å benytte seg av parameterne til et nettverk som allerede har trent på et datasett som utgangspunkt. I mange tilfeller kan dette minke tiden det tar å trene fordi datasettet modellen er trent på er relevant for oppgaven. I tillegg sparer det tid at det kun er det siste FC-laget som trenes, mens resten av nettverket beholder de "fryst" parameterne. *Fine-tuning* benyttes for å spare tid, men også for å kompensere for at treningssettet er for lite til å trene et komplisert dypt nevralt nettverk helt fra bunnen av (Saulo Barreto, 2023).

[2]ILSVRC står for ImageNet Large Scale Visual Recognition Challenge

2.3.8 Dataaugmentering

For å øke størrelsen på treningssettet med hensikt om å få flere datapunkter for modellen å trene på benyttes dataaugmentering *eng. data augmentation*. Ved å øke treningssettet minker man sjansen for en overfit modell. Eksempler på dette ved bruk av bilder er å klippe det på diverse måter og å rotere eller speile dataen. I praksis er dataaugmentering å generere mer data fra den allerede eksisterende og ekte dataen (Géron, 2023, s. 500).

2.3.9 GPU og CPU

GPU (Graphical processing unit) er datamaskinens grafiske prosesseringsenhet og er mest kjent for å gjøre det mulig å fremstille bilder på skjerm. GPU, eller grafikkortet, er designet for å løse mange oppgaver parallelt og består av mange kjerner, ofte tusenvis, som utfører simultane matrisemultiplikasjoner (Caulfield, 2009).

Dype nevrale nettverk krever stor databehandlingskraft fordi det å trene et nettverk innebærer gjentakende matrisemultiplikasjon i store volum. En CPU er i stand til å utføre trening, men bruker betraktelig lenger tid fordi den ikke er utstyrt med nok kjerner som kan utføre oppgaver samtidig. Det å utføre trening på dype nevrale nettverk på GPU har de siste årene vist seg å være revolusjonerende for fagfeltet fordi det har økt kapasiteten for og effektivisert prosessereringen av enorme mengder data (Caulfield, 2009).

2.3.10 Pytorch og CUDA

Denne oppgaven bruker utelukkende Python til å utføre alle operasjoner. Pytorch er et python-bibliotek som er utviklet spesifikt for ML og nevrale nettverk. Den inneholder innebyggede funksjoner og funksjonaliteter som tillater en mer høynivå tilnærming til ML-alogritmer. Pytorch er tilpasset for å gjøre det enklere for programmerere å utvikle ML-script fordi bygningsblokkene for å lage dype nevrale nettverk er innebygd.

CUDA er programvare som er laget for å tillate prosessering på GPU ved at den sørger for at CPU og GPU «snakker sammen» og overfører oppgaver seg imellom. Som nevnt er GPU den vanligste prosesseringsenheten å trene dype nevrale nettverk på i dag, men datamaskiner er forhåndsprogrammert til å utføre alle oppgaver på CPU med mindre noe annet er spesifisert. CUDA gjør at brukeren kan definere hvorvidt

et program skal utføres ved hjelp av GPU og brukes i denne oppgaven til å overføre treningen av et ResNet til GPU (Heller, 2022).

2.4 Validering

Validering innebærer å teste hvor godt modellen presterer. En metode for å gjøre dette er ved å splitte datasettet i to grupper, treningsdata og valideringsdata. Modellen trenes først ved hjelp av treningsdataen og valideres ved å måle hvor god modellen er til å klassifisere valideringsdataen som simulerer ny og usett data. Validering gir et mål på modellens prestasjon på ny data gjøres etter hver epoch for å måle progresjon. Valideringen gir indikasjoner på hvordan modellen presterer og om noe bør justeres for å optimalisere den (Ozechi, 2021).

2.4.1 Læringskurver, Accuracy, Precision, Recall og F1-score

I følgende delkapittel introduseres de relevante verktøyene for å validere en modell, såkalte *performance metrics*. Alle formlene er hentet fra s. 108-113 i Géron, 2023.

Læringskurver

Læringskurver er et verktøy for å se modellens prestasjon underveis som den trenes. To eksempler på læringskurver er å sammenlingne nøyaktigheten og Loss på trenings- og valideringsdataen. Grafene kan gi tydelige indikasjoner på om modellen er overfit eller underfit og viser hvor fort modellen konvergerer mot kostfunksjonens minimum. Et eksempel på læringskurver er vist i figur 3.3.7 i kapittel 3.3.7.

Accuracy

Modellens accuracy gir et helhetlig mål på modellens prestasjon som tar hensyn til alle kategorier. Accuracy er forholdet vist i formel 2.3, der TP er alle prediksjoner som er sann positiv, TN er sann negativ, FP er falsk positiv og FN er falsk negativ. For å få accuracy i prosent ganges resultatet med 100%.

$$Accuracy_{tot} = \frac{TP_{tot} + TN_{tot}}{TP_{tot} + TN_{tot} + FP_{tot} + FN_{tot}} \tag{2.3}$$

Precision

Precision gjør det mulig å se forskjellene mellom kateogoriene. formelen er vist i formel 2.4 . Verdiene inkludere verdiene for kun en kategori eller hele modellen.

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

Recall

Recall ligner veldig på Precision, men måler i stedet forholdet mellom TP og FN som vist i formel 2.5. Verdiene inkludere verdiene for kun en kategori eller hele modellen.

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

F1-score

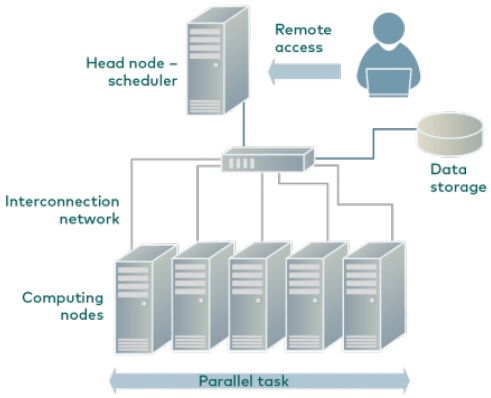
Til slutt introduseres F1-score som er et gjennomsnitt av Precision og Recall. Grunnen for å bruke F1-score er å finne den optimale vektingen mellom Recall og Precision ettersom den ene øker når den andre minker. F1-score regnes ut i formel 2.6.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.6}$$

2.5 Maskinklynge

En maskinklynge *eng. computer cluster og High Performance Computer (HPC) cluster* er et sett med datamaskiner som er koblet sammen i et nettverk som samlet gir høy ytelse og maskinkraft. Klyngen av mange maskiner fremstår for brukeren som én kraftfull datamaskin som kan brukes av flere samtidig og utføre mer krevende oppgaver enn det man kan gjøre på en enkelt maskin (SUSE, 2023).

Hver maskinklynge er ulik fordi det finnes mange kombinasjoner av datakomponenter ut ifra hvilke oppgaver som skal utføres. Det å ha mange datamaskiner tillater også enkel utskiftning av enkeltkomponenter etter hvert som teknologien utvikler seg og behovet endrer seg (SUSE, 2023). En klynge består av flere noder, altså datamaskiner, som er koblet sammen. Maskinklynger er en måte å få høy ytelse som er mer robust



Figur 2.5.1: Diagram av maskinklyngens struktur som viser hovednode, vanlige noder og lagringsplass (Riga Technical University, n.d.).

mot feil, og er designet slik at funksjonelle noder kan fortsette arbeidet selv når andre er satt ut av spill (SUSE, 2023).

En maskinklynge kan brukes av flere brukere samtidig og kan aksesserer den gjennom hovednoden *eng. head node*. For å administrere alle brukernes behov og tilgang til maskinene brukes et *centralized management system*, nemlig at distribusjonen av noder er regulert av programvare som sørger for ryddig kommunikasjon mellom nodene og hindrer feil (SUSE, 2023). Figur 2.5.1 viser en generell struktur bestående av hovednode, andre noder, disker og bruker.

SSH

SSH står for Secure Shell og er software som gjør det mulig å få tilgang til andre datamaskiner via nett på en kryptert og trygg måte (SSH, 2023). SSH benyttes i denne oppgaven til å få tilgang til HPC-serveren fra lokal PC. Et eksempel på bruk er vist i vedlegg B.

Slurm

Slurm (Simple Linux Utility for Resource Management) er et *Workload management system*, et slags køsystem for brukene av maskinklyngen, som er utviklet for å automatisere prosessen for tildeling av ressurser til oppgaver. Den regulerer hvilke oppdrag *eng. jobs* klyngen skal ta for seg og når, ut ifra hvilke ressurser som er tilgjengelige. Det er systemadministrator som bestemmer hvilken *workload manager* som benyttes. I

tillegg til å allokere ressurser til brukernes oppgaver kan køsystemet overvåke pågående prosesser som kjører og har overordnet myndighet til å stanse og prioritere oppgaver etter gitte krav (Bright Computing, 2023, s. 18). Slurm er en av flere workload managers og er mye brukt av maskinklynger rundt i verden.

Hver bruker må be om tillatelse *eng. submit job* for å bruke maskinklyngens ressurser ved å kjøre en "job script". Job scriptet består av diverse informasjon om blant annet hvor mye minne oppgaven trenger, hvor mange GPU-er som avsettes og hvor lang tid oppgaven tar. Workload manageren ser om de nødvendige ressursene er tilgjengelig og sender oppgaven videre idet ressursene tildeles (Bright Computing, 2023, s.18).

Implementering

Dette kapittelet beskriver implementeringen og realisering av forsøkene. Kapittelet går først gjennom hardware, software og datasettene, før den gjennomgår den overordnede fremgangsmåten som er felles for forsøkene. Til slutt forklares de individuelle fremgangsmåtene for forsøk 1 og 2.

3.1 Hardware

En stor del av oppgaven er å håndtere mangelen på tilstrekkelig maskinkraft i lokalt tilgjengelig hardware. Løsningen er å bruke både en lokal bærpar PC og en maskinklynge. All testing, preprosessering og feilsøking ble gjennomført på en bærbar datamaskin utlånt fra IKT på Sjøkrigsskolen. Selve forsøkene og treningen ble utført på den eksterne maskinklyngen *eX3*.

Den bærbare datamaskinen er beskrevet i tabell 3.1.1. Denne maskinen er kategorisert som en standard bruksmaskin som håndterer normale oppgaver som ikke krever stor maskinkraft. Det ble støtt på flere problemer med denne maskinen fordi den ikke var kraftig nok til å gjøre større prosesser, som for eksempel å konvertere 12 000 lydfiler til spektrogrammer. Resultatet av dette var at testing, feilsøking og preprosesseringen tok flere timer enn ønsket. Når treningsscriptet ble testet lokalt tok hver epoch rundt 45 minutter, som gjør det urealistisk å få trent nettverket med titalls epoch.

På grunn av den manglende maskinkapasiteten til å gjennomføre effektiv trening på Sjøkrigsskolen ble den eksterne maskinklyngen *eX3* benyttet. *eX3* er en norsk-etablert HPC-infrastruktur laget av Simula Research Laboratory i samarbeid med partnere fra forskningsmiljøer i Norge. Maskinklyngen har hardware tilpasset mange ulike bruksområder. Oppgaven har benyttet seg av en partition som har 2x Intel Xeon Platinum

Navn:	Lenovo ThinkPad E470
RAM:	8.0 GB
Processor:	Intel® Core™ i7-7500U CPU @ 2.70GHz × 4
Graphics:	NV118 / Mesa Intel® HD Graphics 620 (KBL GT2)
Disk:	256 GB

Tabell 3.1.1: Tabell som inneholder teknisk informasjon om lokal PC benyttet i oppgaven.

8186 CPUer , 16x NVIDIA Tesla V100 GPUer 12 integrerte NVS-switcher. Totalt har eX3 500 TB med lagringskapasitet (eX3, 2023).

3.2 Software

Delkapittelet lister opp og beskriver software som har blitt benyttet i oppgavens problemløsning. Hovedspråket benyttet er Python og operativsystemet benyttet er Ubuntu 22.04.3 LTS.

Python er et objektorientert høynivå programmeringsspråk som brukes over hele verden til et bredt spekter av bruksområder. Språket er gratis og open-source (Python Software Foundation, n.d.). Språket har blitt valgt i denne oppgaven med hensikt om å lære det, og å kunne benytte seg av alle ressursene som eksisterer på nett for feilsøking. I tillegg oppleves språket enkelt å lære seg, derav overkommelig for et prosjekt med relativt kort tidslinje. Python er enkel å installere og har en stor fordel av å være kompatibel med et stort antall biblioteker som kan lastes ned. Tabell 3.2.1 inneholder en liste av alle bibliotekene som har blitt brukt i denne oppgaven.

3.3 Data

For å sikre gode forutsetninger for å få pålitelige resultater er det viktig å oppdrive data som er så relevant som mulig til nettverkets bruksområde og som finnes i store datasett. Ideelt bør denne dataen bestå av reell data fra den aktuelle hydrofonen eller sonaren

Bibliotek	Beskrivelse
NumPy	Det grunnleggende biblioteket for vitenskapelig programering i Python. Det inneholder diverse matematiske funksjoner som blant annet muliggjør Fouriertransformasjoner og plottingen av disse. Det er gjennom disse utregningene at spektrogrammene blir utvinnet (NumPy Developers, 2023).
SciPy	Utvidelse av NumPy som innehar algoritmer til bruk for mer avanserte matematiske operasjoner som lineæralgebra. I denne oppgaven brukes SciPy til å hente ut dataen fra wav-filer.
Pydub	Utviklet for å jobbe med .wav-filer. Det brukes i scriptet i vedlegg A.2 til å splitte, klippe og formatere wav-filer.
PIL	Bibliotek designet for bildeprosessering som brukes til å endre størrelse på spektrogrammene slik at dimensjonene er tilpasset nettverkets inngangsformat. PIL står for Python Imaging Library.
Matplotlib	Bibliotek som lager visualiseringer av data og muliggjør plotting i alle slags former. Den inneholder funksjonene som plotter spektrogrammene og diverse plotter i oppgaven.
Shutil	Designet for å gjennomføre operasjoner på filer og mapper. Det er et supplement til standard-biblioteket os. Shutil brukes til flytting av filer mellom mapper.
Pytorch	Maskinlæringsbibliotek og beskrevet i kapittel 2.3.10.

Tabell 3.2.1: Tabell med informasjon om de viktigste Python-biblioteker benyttet i implementeringen.

som er tiltenkt å bli benyttet til systemet, men i en militær kontekst er slik data stort sett gradert og utilgjengelig for testing. Sitatet:

> *The mapping function learned will only be as good as the data you provide it from which to learn,* (Jason Brownlee, 2019)

beskriver at dataen må inneholde kjennetegnene som karakteriserer mønsteret systemet skal generalisere.

Størrelsen på datasettet avhenger av kompleksiteten på problemstillingen og nettverkets kompleksitet. Måten å finne ut om man har tilstrekkelig mengde med data er gjennom testing (Jason Brownlee, 2019). Størrelsen på datasettet avhenger av oppgavens kompleksitet, som for denne oppgaven betyr at det trengs *en del data,* gjerne flere tusen eksempler.

3.3.1 Primærdatasett- DeepShip

Datasettet er hentet fra samarbeidsprosjektet Deepship (Irfan et al., 2021) mottatt via FFI. Dette datasettet består av rundt 50 timer med annoterte opptak av skipspasseringer. Gjennom dokumentet vil datasettet bli referert til som *DeepShip*. Fordelen med datasettet er størrelsen, som overgår andre kjente annoterte datasett av fartøyspasseringer. I tillegg inneholder DeepShip opptak fra alle årstider og mange værtyper, der datasettet i stor grad representerer varierende omgivelser, med unntak av at dataen har opprinnelse fra den samme statiske hydrofonen.

Rådataen til Deepship er samlet inn av Ocean Networks Canada ved hjelp av en hydrofon lokalisert sør-vest for Vancouver i Georgiasundet i Canada på 170-meters dybde (Ocean Networks Canada, 2023). Råopptakene er tatt i tre separate perioder mellom mai 2016 og oktober 2018 resulterende i rundt 17 måneder med råopptak (Irfan et al., 2021). Rådataen har deretter blitt annotert av Irfan et al. (2021) ved å sammenligne AIS-data (automatic identification system) for enkeltfartøy som har passert hydrofonen innenfor en 2km radius med lydklippene for den samme tidsperioden. Ved å bruke AIS-data har Irfan et al. (2021) generert et stort datasett med opptak av fartøyspasseringer av 256 unike fartøy kategorisert i fire klasser: *Cargo*, *Passengership*, *Tanker* og *Tug*.

Hydrofonen ligger i en en av Canadas mest trafikkerte leder og er preget av støy fra både skipstrafikk og biologisk aktivitet. Farvannet har mange elveutløp som påvirker strømninger i tilleg til tidevannsstrøm. Havbunnen består av silt og sand (Irfan et al., 2021, s. 4). Lokasjonen til hydrofonen er markert med hjertet i figur 3.3.1.

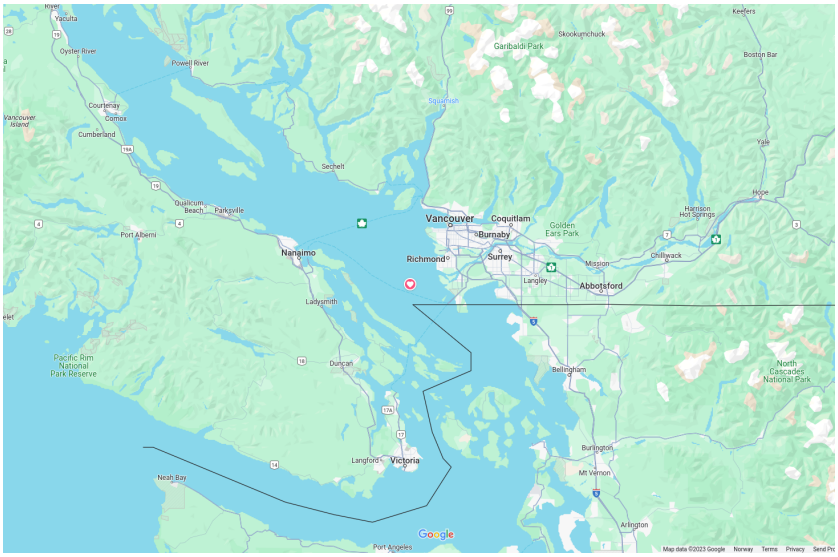
Datasettet består av til sammen 47 timer og 4 minutter med lydklipp i .wav-format. Lydklippene varierer i lengde mellom 10-400 sekunder. Hydrofonen har en samplingsrate på 32kHz. I tabell 3.3.1 vises antall sekunder med opptak per klasse.

Datasettet inneholder metafiler for hver kategori som lister opp diverse data punkter relatert til hvert lydklipp. Denne informasjonen benyttes ikke i denne oppgaven, men kan være av relevans. Metadataen inneholder følgende informasjon i hver passering:

> 78 - 70 - GALLEON - 20171206 - 132824 - 194 - 711 - 1054

Som står for:

> ID - Class ID (MMSI) - navn på skip - dato - tid - lengde på klipp - distanse min i meter, distanse maks i meter



Figur 3.3.1: Kart av geografik lokasjon til hydrofonen som har innhentet dataen i DeepShip-datasettet.

Klasse:	Cargo	Passasjer	Slepebåt	Tankbåt
Lengde(s):	38 400	44 520	40 620	45 900

Tabell 3.3.1: Antall sekunder med markert data per kategori (Irfan et al., 2021).

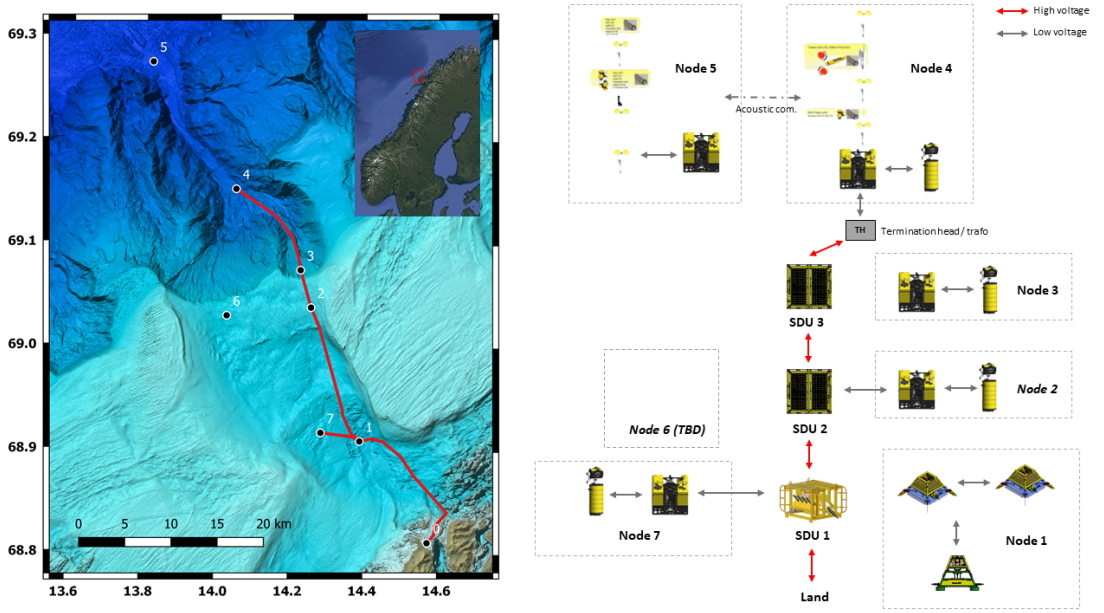
Hydrofonen som har tatt opp lyden er en Ocean Sonics icListen AF som har en båndbredde mellom 1Hz og 12kHz (Irfan et al., 2021, s. 4).

3.3.2 Sekundærdatasett - LoVe

LoVe-data er hentet fra Lofoten-Vesterålen Ocean Observatory (LoVe Ocean). I implementeringen er volumet av data fra Love betraktelig mindre enn fra DeepShip. Dataen fra LoVe benyttes utelukkende som testdata. LoVe-datasettet er laget og tilsendt av Lars Alf Ødegaard fra Forsvarets Forskningsinsituttt.

LoVe Ocean er et observatorium bestående av et nettverk av sjøkabler som overvåker undervannsmiljøet nord for Lofoten og Vesterålen. Ved hver node er det diverse vitenskapelige sensorer, noen av dem statiske hydrofoner som innhenter lyddata. Rådataen er hentet fra LoVe, men har blitt prosessert av FFI der lyfilene er delt inn i enkeltpasseringer.

Dataen er produsert av en hydrofon tilkoblet LoVe Node 1 som ligger på 250 meters dyp. Undervannsmiljøet hvor hydrofonene er plassert bærer preg av mangfoldig biologisk aktivitet, sedimentbunn og store dybdevariasjoner (Elle, 2017). Noden er plassert



Figur 3.3.2: Diagram hentet fra LoVeOcean, 2023 som viser lokasjon på nodene og infrastrukturen plassert ut.

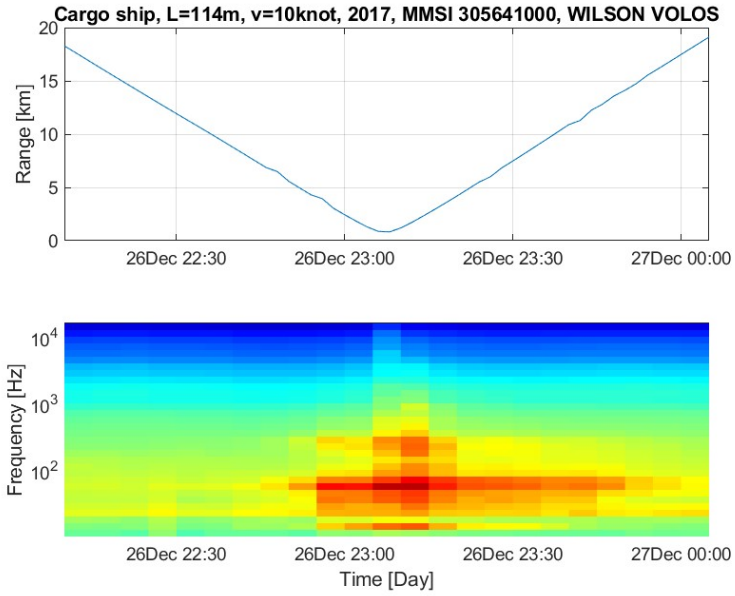
på relativt grunt vann, men er omringet av større og mindre dybdeendringer som kan bidra til mer ekko og støy. Figur 3.3.2 viser LoVe sine noder og deres relative plassering.

Datasettet inneholder .wav-filer på 10 minutter som dekker tidsrommet fartøyene passerer hydrofonen. Dataen er ikke direkte annotert, men er sendt med informasjon i form av avstand/tid-grafer som viser AIS-data og et spektrogram av tiden lydklippene dekker. Grafen viser avstander mellom hydrofon og skip fra ca. 18km og nærmere. Et eksempel er vist i figur 3.3.3.

3.3.3 Valg av nettverk

Oppgavens problemstilling krever et nevralt nettverk som er designet for gjenkjenning av bilder. For å gi en solid og forutsigbar ramme til forsøkene har valget falt på å benytte publiserte og godt testede dyplærings-arkitekturer som er designet for å fungere godt med grafisk input.

I forskningen til Irfan et al., 2021, kom ResNet på andreplass av fem kjente dype nevrale nettverk. Dette, i kombinasjon med at ResNet er brukervennlig og godt dokumentert gjorde at denne nettverkstypen ble valgt som testplattform. Dessuten er ikke oppgavens mål å finne fram til hvilket nettverk som presterer best i nøyaktighet, men



Figur 3.3.3: Datainformasjon som mottatt av Lars Alf Ødegaard. Øvre graf viser avstand til hydrofon over tid, nedre graf er et spektrogram av passeringen.

i stedet å sammenligne prestasjonen til ett nettverk i variable miljøer. Etter anbefaling fra veileder og basert på at ResNet er anerkjent for å prestere godt med bilder, ble dette valgt som nettverk.

Det finnes flere typer ResNet basert på hvor mange lag nettverket består av. I denne oppgaven benyttes ResNet-18 fordi det var mest tilgjengelig. Mer informasjon om ResNet er nevnt i kapittel 2.3.6.

3.3.4 Fremgangsmåte

Delkapittelet tar for seg realiseringen av fremgangsmåten steg for steg. Hensikten er å forklare implementeringen i detalj for å bygge oppunder resultatene og å vise til steg i prosessen som påvirker disse.

Alle script er enten generert og hentet fra OpenAI sin ChatGPT eller fra nettsider som GitHub. Kode hentet fra KI-ressurser og nettbaserte kilder er modifisert og tilpasset oppgaven. Implementering av kode fra andre kilder har gitt stor tidsbesparelse og muligheten til å fokusere på andre, mer lærerike aspekter ved oppgaven, som bruk av *eX3* og Pytorch.

3.3.5 Preprosessering

I implementeringen består preprosesseringen av følgende steg:

1. Klipping av lydfiler til 3 sekunder.
2. Konvertering til spektrogrammer
3. Reskalering av data til 224×224 piksler
4. Fordeling til trenings- og valideringssett

Fremgangsmåten for forsøk 1 og 2 er identiske for steg 1,2 og 3 fordi dataen er prosessert helt lik i begge forsøkene.

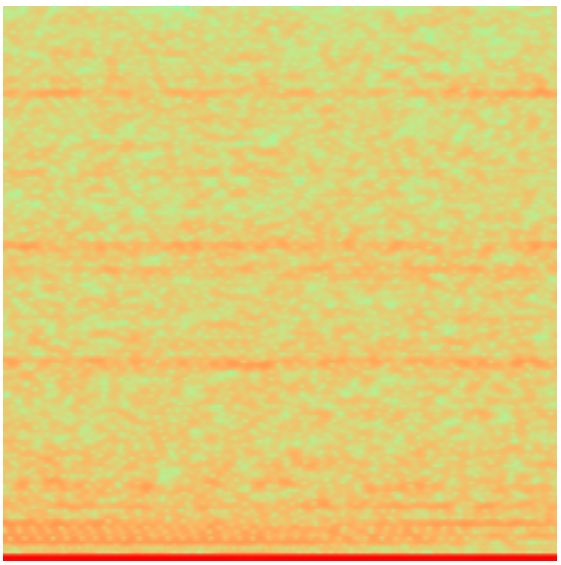
1. Klipping av lydfiler

Råe lydfiler kan ha vilkårlige tidslengder, som gjør dem ugunstige å bruke til å lage spektrogrammer som skal ha like verdi- og definisjonsmengder. I tillegg er tidsintervallene nokså store, slik at kjennetegn i dataen kan forsvinne blant alle datapunktene. Det å dele opp lydfilene i mange små klipp vil øke mengden data som modellen trenes på og er et forsøk på dataaugmentering.

For å øke mengden treningsdata blir hvert lydklipp delt opp mindre klipp på tre sekunder. Dette er et forsøk på å få en relativt høy multiplikasjon av datapunkter, samtidig som mønstrene i dataen holdes intakt. Det er ingen fasit på hvilken lengde lyddataen bør være. Kortere klipp betyr flere datapunkter, men i utgangspunktet må denne variabelen testes for å finne den optimale kombinasjonen. Scriptet i vedlegg A.2 brukes til å dele hvert originale lydklipp inn i klipp på 3 sekunder. Navngivning av klippene og det å ha oversikt over mappene er viktig fordi det fort går fra å være få klipp til å bli flere titalls tusen.

2. Konvertering til spektrogrammer

Datasettet er opprinnelig en lydfil i wav-format og må konverteres til visuelle data i form av spektrogrammer i png-format. I samme prosess er det viktig at det essensielle i dataen kommer fram slik at modellen får bedre forutsetning for å generalisere dataens



Figur 3.3.4: Spektrogram fra kategorien *Tug*. Y-aksen går mellom 0-1,250kHz og x-aksen mellom $(0, 3)$s.

kjennetegn. Scriptet i vedlegg A.3 itererer gjennom alle lydklipp og lager spektrogrammer ved hjelp av STFT. Amplituden til frekvensene i lydfilene blir plottet etter en fargekode der mørkerøde piksler representerer høy amplitude.

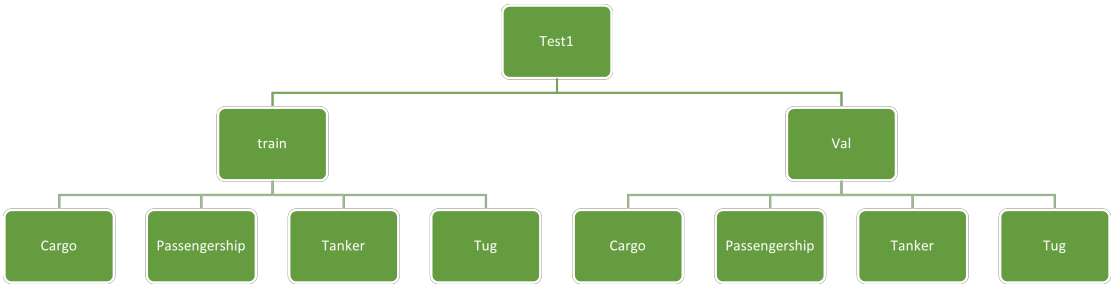
Det er hovedfrekvensene som definerer hvilke frekvenser som bør plottes. I skipsfart er det som nevnt de nedre frekvensene mellom 50-500Hz. Hvis skalaen ikke er tilpasset frekvensene av interesse kan det være umulig å tyde mønstrene i dataen. I et forsøk på å fremheve de relevante frekvensene i lydfilene ble spektrogrammene plottet mellom 0 og 1,250 kHz. På denne måten ble avstanden mellom hovedfrekvensene større og derav mer synlige for både øyet og for algoritmen. Kodeeksempel 3.1 viser linjen i scriptet i vedlegg A.3 som endrer verdimengden. Figur 3.3.4 viser et spektrogram fra en passering fra kategorien *Tug* i DeepShip.

,

```
1 import matplotlib.pyplot as plt
2 plt.ylim(0, 1250)
```

Kodeeksempel 3.1: Metode for å endre maksimale y-verdi, i dette tilfellet enheten frekvens (Hz).

I tillegg til endring av verdimengden ble fargekode og n_{fft} endret. Flere detaljer om konvertering til spektrogram ligger i vedlegg E.



Figur 3.3.5: Tre-diagram av mappestrukturen til datasettet. Hver grønn firkant er markert med navn og representerer mapper.

3. Reskalering av data til 224×224 piksler

ResNet-18 begrenser hvilke dimensjoner png-filene kan være for at nettverket kan benytte filen til trening. Konvulsjonsnett behandler hver piksel i bildet, som gjør at for store bilder senker læringstiden. ResNet er designet for å ta i mot filer på 224×224 piksler. For å reskalere bildene til riktig dimensjon benyttes Python-biblioteket Pillow og funksjonen *resize()*. Å reskalere bilder er en teknisk avansert operasjon som ikke vil forklares i dybden i denne oppgaven. Metoden for reskalering er vist i vedlegg E.

4. Fordeling mellom trenings- og valideringsdata

For at nettverket skal klare å ta imot og prosessere dataen må den være tilpasset og kategorisert i den strukturen som er spesifisert av nettverket. Det viktigste er at datasettet er fordelt i mapper som er navngitt på riktig måte og plassert på riktig sted. Figur 3.3.5 viser at datasettet er fordelt i et hierarki først bestående av *train* og *val* før hver mappe inneholder mapper tilsvarende de fire kategoriene. For at valideringssettet skal gi pålitelige resultater er det viktig at dataen forbeholdt validering er representative av treningsdataen uten å ha for stor bias. Data som er biased kan bety at det ligner altfor mye på datapunktene i treningssettet. Dette kan forsøkes å unngås ved at klipp på tre sekunder med lik opprinnelse alltid fordeles til enten train eller val. Dataen fordeles slik at ca. 75% av dataen ligger i treningssettet og ca. 25% i valideringssettet. Det viktigste er å sørge for at mappenavnene i "train" og "val"-mappe tilsvarer klassene og er identiske i både trenings- og valideringsmappene.

Antall spektrogram fra DeepShip i hver kategori i forsøk 1			
Kategori	train	val	totalt
Cargo	10 019	3785	13 804
Passengership	11 873	3532	15 405
Tanker	11 632	3128	14 760
Tug	9708	2782	12 490

Tabell 3.3.2: Tabell av antall spektrogrammer i hver kategori i train- og valideringssettet for forsøk 1.

Antall spektrogram fra DeepShip i hver kategori i forsøk 2			
Kategori	train	val	totalt
Cargo	8806	3995	12 801
Passengership	2592	856	3412
Tanker	2543	820	3363
Tug	10 380	3113	13 493

Tabell 3.3.3: Tabell av antall spektrogrammer i hver kategori i train- og valideringssettet for forsøk 1.

Treningsdata

I forsøk 1 trenes et nettverk med all data fra DeepShip. Antallet png-bilder i hver kategori i trenings- og valideringssettet er vist i tabell 3.3.2.

I forsøk 2 er kun DeepShip-data med opprinelse fra vinterhalvåret er inkludert i trenings- og valideringssettet. Måneder som regnes som vinterhalvåret er november, desember, januar, februar og mars. Filene er navngitt med datoen de har opprinnelse fra, som gjør det mulig å ekstrahere dataen fra DeepShip-datasettet.

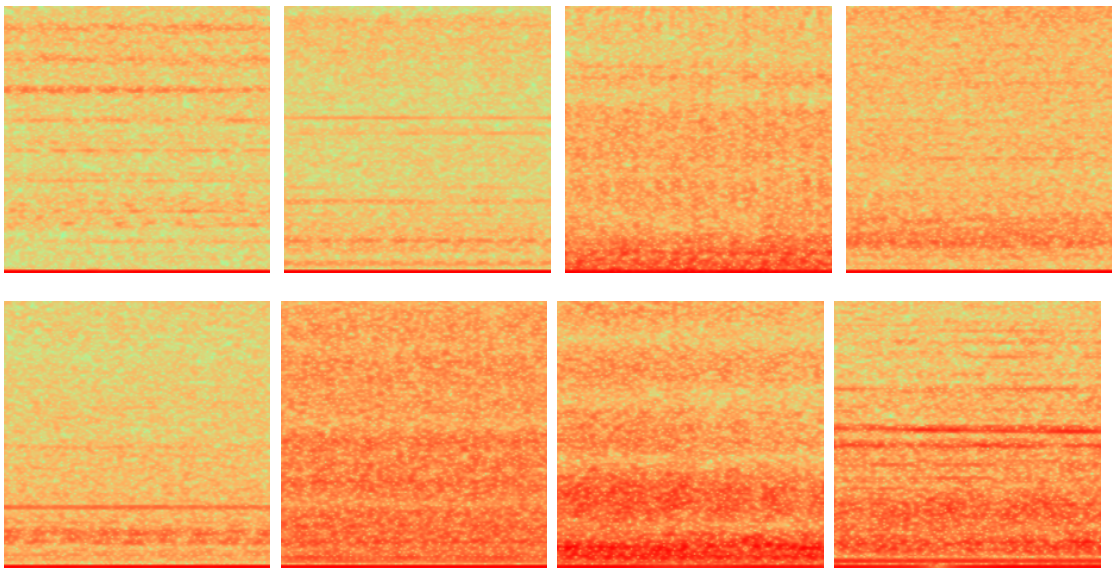
Spesielt *Passengership* og *Tanker* hadde relativt få eksempler fra vinterhalvåret, og spesielt *Tug* hadde bare data fra vinterhalvåret. Antall bilder i hver kategori i forsøk 2 er vist i tabell 3.3.3.

3.3.6 Testdata

Testadataen defineres som data uavhengig av trenings- og evalueringssettene som modellen skal testes på. LoVe-datasettet benyttes kun som testdata, der dataen er prepros-essert med lik metode som DeepShip-datasettet. Hensikten med dette er å gjøre formateringen av testdata og trenings- og valideringsdata så like som mulig, helst identisk. På

den måten er datasettene mer sammenlignbare.

I forsøk 1 benyttes LoVe- data som testdata, der minimum ett eksempel per kategori er preprossesert for å teste hvor godt nettverket er til å klassifisere LoVe-dataen. Tabell 3.3.4 inneholder detaljert informasjon om testdataen for forsøk 1 og 2. For forsøk 1 sammenlignes data fra to ulike datasett. I figur 3.3.6 vises fire eksempler fra testdataen i nedre rad sammenlignet med treningsdata i øvre rad. LoVe-datasettet inneholder 2 passeringer for kategorien *Cargo*, og 2 passering for *Passengership*, *Tanker* og *Tug*.



(e) Cargo **(f)** Passengership **(g)** Tanker **(h)** Tug

Figur 3.3.6: Eksempler fra hver kategori i treningsdata fra DeepShip (øvre) og testdata fra LoVe (nedre) for sammenligning.

I forsøk 2 benyttes ferdigprosesserte bilder med opphav fra sommermånedene i DeepShip-datasettet som testdata. Fordi det er færre eller null eksempler fra sommerhalvåret innenfor to av kategoriene uteblir disse fra testdataen. Tabell 3.3.4 inneholder detaljert informasjon om testdataen for forsøk 1 og 2. Tabellen viser også at totalt antall spektrogrammer i kategorien *Tug* i forsøk 2 enn i forsøk 1. Antakeligvis er dette grunnet en feilfordeling som kan ha innvirkning på resultatet.

3.3.7 Trening

Etter at dataen er preprosessert kan nettverket trenes og testes. Dette delkapittelet gjennomgår realiseringen av nettverkets treningsprosess.

Scriptet benyttet til konstruksjon og trening av nettverket er laget av Sasank Chil-

Datapunkter i testdata for forsøk 1 og 2		
Kategori	forsøk 1	forsøk 2
Cargo	240	0
Passengership	120	925
Tanker	200	810
Tug	200	0
Totalt:	750	1735

Tabell 3.3.4: Tabell av antall testdata i forsøk 1 og 2.

amkurthy og hentet fra PyTorch sine nettsider (Sasank Chilamkurthy, 2023). Originalkoden er skrevet i som en Jupyter Notebook [1] og er modifisert til å plotte læringskurver. Scriptet er skrevet i Python og bruker Pytorch-biblioteket. Kapittelet vil ikke gå i dybden på spesifikk syntaks for programmering innenfor maskinlæring.

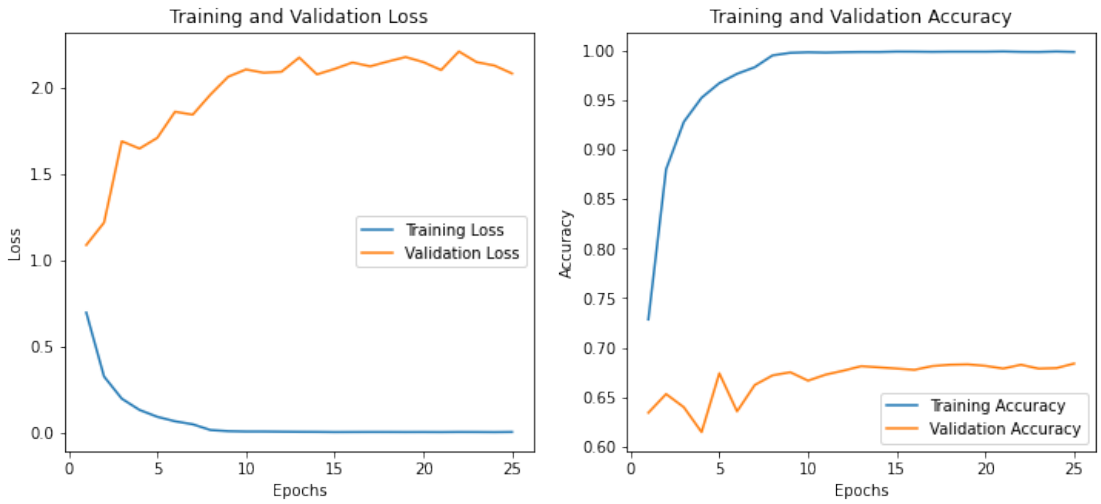
For å komme til steget der nettverket trenes må scriptet og sbatch-filen klargjøres og overføres til maskinklyngen. Hvert forsøk har sitt script som inkluderer sin respektive treningsalgoritme. Disse er helt like for begge forsøk, foruten kodelinjen med referanse til riktig treningsdata. Treningsscriptene har blitt modifisert til å plotte læringskurver.

Scriptet inneholder en treningsalgoritme som fine-tuner et ResNet-18-nettverk som er forhåndstrent på ImageNet[2]. Hver epoch er delt inn i en treningsfase og en valideringsfase. I treningsfasen itererer modellen gjennom treningsdatasettet og optimaliserer parameterne med backward propagation og gradient descent. Til slutt tester modellen hvor godt den predikerer testdataen og gir et mål på *Training Loss* og *Training Accuracy*. I valideringsfasen går modellen gjennom valideringsdataen og tester modellens parametere på ny data for å få en verdi på *Validation Loss* og *Validation Accuracy*. Scriptet ligger i sin i helhet i vedlegg A.6. Hver epoch gir output:

- **Training Loss:** Loss-verdi for treningsfasen. Sier noe om output til kostfunksjonen som forteller hvor stor feil modellen har.
- **Training Acc:** Accuracy etter testing på treningsdata
- **Val Loss:** Loss-verdi for valideringsfasen. Forteller output til kostfunksjonen i

[1] Jupyter Notebook en interaktiv utviklingsplattform som bl.a. gjør det mulig å kjøre kodesnutter i web-browser (Antonino Ingargiola et al., 2015).

[2] ImageNet er et datasett bestående av hundretusener av bilder delt inn i kategorier med ca. tusen bilder i hver kategori. Datasettet er open-source og brukes til forhåndstrening av nettverk(Stanford Vision Lab, 2020).



Figur 3.3.7: Læringskurver av Training og Validation Loss over epochs til venstre og Training og Validation Accuracy til høyre. Grafene viser modellens progresjon gjennom treningen.

```
Training complete in 73m 36s
Best val Acc: 0.686134
```

Figur 3.3.8: Skjermbilde av output etter alle epochs er gjennomført. Scriptet oppgir tid brukt på trening og treningens høyeste verdi for Validation Accuracy.

valideringsfasen.

- **Val Acc:** Accuracy etter testing på valideringsdata

For hver epoch lagres de ovennevnte verdiene i hver sin liste for så å bli plottet i hver sine grafer, også kalt læringskurver. Et eksempel i figur 3.3.7 viser læringskurvene for Training og Validation Loss til venstre og Training og Validation Accuracy til høyre. både Loss-verdier for trening- og valideringsresultatene er vist over antall epoch. Figur 3.3.8 viser output etter hver epoch. Den høyeste verdien for Validation Accuracy lagres underveis og angis på slutten av treningen.

For å trene modellen må treningsscriptet kjøres i *eX3* gjennom slurm. Detaljert fremgangsmåte for bruk av ekstern maskinklynge ligger i vedlegg B. For å få allokert ressurser i maskinklyngen er det pålagt å kjøre et sbatch-script. Scriptet skal inneholde essensiell informasjon og inneholder kommandoene som skal kjøres av noden når den er tilgjengelig for bruk. Sbatch-filene for forsøk 1 og 2 ligger i vedlegg A.

For å kjøre en Jupyter Notebook-fil, også kalt .ipynb, uten å kjøre den i browser må den først konverteres til en .nbconvert-fil. Når filen er ferdig kjørt konverteres den

tilbake til en .ipynb-fil og lagres som en .nbconvert.ipynb. Kodeeksempel 3.2 i script i vedlegg A.4 viser kodelinjen som kjører en .ipynb-fil uten browser. Denne linjen må legges inn i .sbatch-filen for å kjøre i *eX3*.

,

```
srun jupyter nbconvert --to notebook --execute
    TransferLearning_script.ipynb
```

Kodeeksempel 3.2: Kommando for å konverte .ipynb-fil til .nbconvert-fil som deretter kjøres.

Når data og alle script er klare for forsøkene gjenstår det å overføre alle filene til eX3-serveren og kjøre .sbatch-filene. Det er mulig å gjennomføre flere Jobs samtidig. Maskinklyngen bruker mellom 60 og 110 minutter på å trene nettverket ved bruk av GPU. Noden benyttet i denne oppgaven har høy kapasitet og er stort sett ledig for middels krevende ML-oppgaver.

Når modellen er trent evalueres den ved hjelp av læringskurvene som gir informasjon om nettverkets treningsutvikling. Evalueringen sier noe om eventuelle endringer som bør gjøres med nettverket for at det skal prestere bedre.

3.3.8 Testing

Det er kun kjøring av treningsscriptet som krever maskinkapasiteten til *eX3* fordi den itererer gjennom store datasett. Siden treningsscriptet er modifisert til å lagre parametrene i en .pt-fil som skrives til *eX3*-serveren kan denne overføres til lokal PC hvor testene kan gjennomføres. Kodeeksempel 3.3 viser kodelinjene som lagrer modellen som en .pt-fil og laster den inn igjen et annet sted.

```
model_scripted = torch.jit.script(model_ft) # Eksporter til
    TorchScript
model_scripted.save('model_scripted.pt') # Lagrer modell som scripted
     model
```

Kodeeksempel 3.3: Kommandoer for å lagrer modellen som .pt-fil.

For å få tilgang til filen på lokal PC brukes SCP til å hente filen fra *eX3*-serveren. Metoden for å lagre en PyTorch-modell ble hentet fra Matthew Inkawhich, 2023. I tillegg til modellen bør .nb-convert.ipynb-filen overføres til lokal PC for å lese av resultatene av treningsprossesen. Når modellen er tilgjengelig brukes scriptet i vedlegg

A.5 til å iterere gjennom alle testdataen for å få prediksjoner. For å vurdere nettverkets prestasjon klassifiserer scriptet testdataen ved bruk av den lagrede modellen og oppgir hvilke prediksjonsverdier den får for hver av kategoriene. Kategorien som får høyest prediksjonsverdi blir den endelige prediksjonen av bildet. I tillegg til prediksjonsverdi for hver kategori er scriptet modifisert til å oppgi TP og FN.

Resultater

Kapittelet presenterer forsøkenes resultater. Etter at treningsscriptet har blitt kjørt i maskinklyngen blir resultatene skrevet til en .nbconvert.ipynb-fil og avlest i browser på lokal PC. Resultatene blir presentert separat for hvert forsøk.

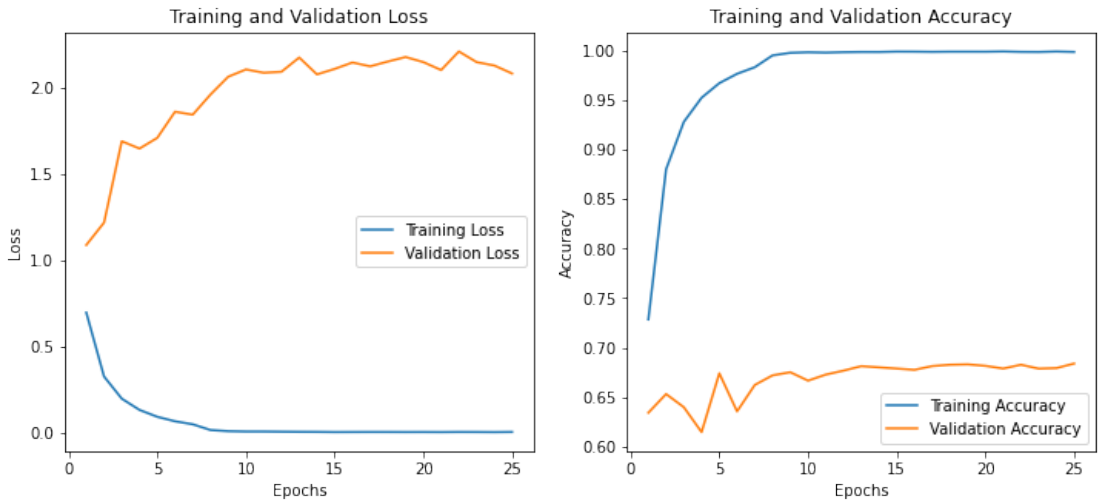
4.1 Forsøk 1 - Sammenligning av DeepShip og LoVe

Treningen av nettverket tok 119 minutter og 56 sekunder. Den korte tiden skyldes at nettverket fine-tuner det ytterste *dense*-laget i stedet for hele nettverket. I tillegg har nettverket benyttet GPU til prosesseringen fordi den innebærer iterasjon gjennom over femti tusen bilder på 224×224 piksler.

I tabell 4.1.1 vises både tiden nettverket bruker på å trene og nettverkets høyeste Val Accuracy. Denne verdien forteller hvor godt nettverket evner å klassifisere spektrogrammene i valideringssettet. Nettverket trent i forsøk 1 fikk sin høyeste verdi på **68,38%**. I figur 4.1.1 vises læringskurver til nettverkt, der det på venstresiden er plottet Loss over epochs og på høyresiden plottet Accuracy over epochs. Loss-grafen viser tydelig hvordan Training Loss stabiliserer seg på en veldig lav verdi allerede etter 10 epochs. På samme måte stabiliserer Training Accuracy seg på rundt 99,99% etter 10 epochs. Validation Loss og Accuracy er derimot mindre stabile men holder seg på et jevnt nivå etter 10 epochs.

Resultater Forsøk 1	
Tid:	119m 56s
Max Val Accuracy:	68.38%

Tabell 4.1.1: Tabell som viser tid brukt til trening og verdien for Max Validation Accuracy i forsøk 1.



Figur 4.1.1: Læringskurver for forsøk 1. Graf til venstre viser Training og Validation Loss over epochs. Graf til høyre viser Training og Validation Accuracy over epochs.

Resultater Forsøk 1				
Kategori	Accuracy	Precision	Recall	F1-score
Cargo		24.14%	34.57%	28.43%
Passengership		9.09%	4.56%	6.07%
Tanker		63.68%	37.98%	47.58%
Tug		0.00%	0.00%	0.00%
Hele modellen:	25.83%	26.00%	27.90%	26.91%

Tabell 4.1.2: Tabell av testresultater som viser Accuracy, Precision, Recall og F1-scor for forsøk 1.

Som vist i Loss-grafen til venstre i figur 4.1.1 øker Validation Loss i løpet av de første 10 epoch-ene. Dette er et tegn på at nettverket presterer stadig dårligere på valideringsdataen. Det samme gjelder ikke for Validation Accuracy, som har en svak økning de første 10 epoch-ene før den stabiliserer seg på rundt 68%.

Etter treningen er gjennomført testes nettverket på testdata som består av bilder som ikke er i trenings- og valideringssettene. Testadataen i forsøk 1 har opprinnelse fra LoVe-datasettet som skal simulere en endring i geografisk lokasjon og sensor. Tabell 4.1.2 viser resultatene til testene i forsøk 1. Modellen fikk en Accuracy på 25.83%. Verdien for Precision, Recall og F1-score for hver kategori er også oppgitt i tabellen. Modellen har en Precision på 26.00%, Recall på 27.90% og F1-score på 26.91%. Mer detaljert fremlegging av resultatdata ligger i vedlegg C.

Resultater Forsøk 2	
Tid:	**73m 36s**
Max Val Accuracy:	**68.61%**

Tabell 4.2.1: Tabell som viser tid brukt til trening og verdien for Max Validation Accuracy i forsøk 2.

Figur 4.2.1: Læringskurver for forsøk 2. Graf til venstre viser Training og Validation Loss over epochs. Graf til høyre viser Training og Validation Accuracy over epochs.

4.2 Forsøk 2 - Sammenligning av årstider

Trening på nettverk 2 tok 73 minutter og 36 sekunder, som er litt kortere enn i forsøk 1, antakeligvis fordi nettverket skal trenes på færre bilder, totalt rundt tretti tusen bilder. Tabell 4.2.1 angir tid og den maksimale verdien på Val Accuracy. Etter 25 epochs er den høyeste Val Accuracy på **68,61%**.

Tabell 4.2.1 er læringskurvene plottet på samme måte som i forsøk 1. Training Loss og Training Accuracy jevner seg ut etter rundt 10 epochs og havner på svært lav Loss-verdi og svært høy Accuracy-verdi etter under 10 epochs. Tabell 4.2.1 viser i tillegg at kurvene for Validation Loss har en økning i løpet av treningen.

For å teste nettverkets evne til å klassifisere data i forskjellige årstider DeepShip-testdata med passeringer fra sommerhalvåret brukt til å testing. Resultatet er vist i tabell 4.2.2. Nettverket fikk en Accuracy på 29.80% for alle kategorier. Tabellen inkluderer kun verdier for de to kategoriene *Passengership* og *Tanker* som ble testet. Modellen har en Precision på 91.68%, Recall på 22.58% og F1-score på 20.52%.

Resultater Forsøk 2				
Kategori	Accuracy	Precision	Recall	F1-score
Passengership		95.65%	11.87%	21.20%
Tanker		87.70%	33.28%	48.25%
Hele modellen:	29.80%	91.68 %	22.58%	36.24%

Tabell 4.2.2: Tabell av testresultater som viser Accuracy, Precision, Recall og F1-scor for forsøk 2.

Drøfting

I dette kapittelet gjennomgås og drøftes resultatene presentert i kapittel 4. Drøftingen legges fram i to hoveddeler bestående av evaluering av nettverkene og analyse av testresultatene. Alle resultater er samlet i vedlegg C. Val Accuracy (Val Acc) er modellens nøyaktighet i valideringsfasen, Train Accuracy (Train Acc) er modellens nøyaktighet når testet på treningsdataen. Val og Train Loss er verdien for kostfunksjonene i henholdsvis valideringsog treningsfasene. I evalueringen benyttes målene på Precision, Recall og F1-score som er oppgitt i kapittel 2.4.1.

Val Accuracy til begge nettverk er rundt 68% og Train Accuracy rundt 100%, som impliserer at nettverkene i begge forsøk er overfit. Dette er fordi Train Accuracy er urealistisk høye. Test-resultatene gir en test-Accuracy på 25.83% i forsøk 1 og 29.80% i forsøk 2. Disse resultatene tilsier at modellen har prestert relativt lavt på klassifiserering av testdataen. Resultatene og eventuelle forbedringer som kan gjøres blir diskutert videre i kapittelet.

5.1 Evaluering av nettverkene

Under treningen var nettverkene i stand til å predikere korrekt kategori i ca. 68% av tilfellene i både forsøk og 1 og 2. Oppgavens metode tar utgangspunkt i Irfan et al., 2021 sine forsøk som tester fem ulike dyplæringsalgoritmer, der en av dem er ResNet. Preprosesseringen i Irfan et al., 2021 er forskjellig ved at de bruker Mel-Spectrogram som metode for å konvertere lyddata til bilder. I deres resultater klassifiserer et ResNet trent på Mel-Spectrogram riktig i 69.29% av tilfellene, som er sammenlignbart med nettverkene i denne oppgaven som predikerer riktig i 68% av tilfellene (Irfan et al., 2021, s. 9). Sammenligningen gir en bekreftelse på at denne oppgavens modeller

er nærme grensen for hva ResNet kan prestere med DeepShip-datasettet. Resultatene sammenlignes for å bekrefte at metoden benyttet i forsøkene gir lignende resultater som Irfan et al., 2021.

Nettverkenes lave Val Acc og høye Train Acc-verdier gir tydelige indikasjoner på at begge nettverk er overfit. En sterk indikasjonen på dette er at Training Loss går mot null, noe som betyr at modellen evner å klassifisere nærmest 100% av treningsdataen, et tydelig tegn på at modellen memorerer treningsdataen. En annen indikator er at Validation Loss øker jevnt etter hvert som modellen trenes, som kan sees i figur 4.2.1 og 4.1.1. Det at Validation Loss øker og Training Loss går mot null er tydelige tegn på en overfit modell. Ideelt sett skal Validation Loss synke i tråd med Training Loss fordi modellen i teorien skal bli bedre til å klassifisere ny data jo mer den trenes. I forsøk 1 og 2 blir stadig bedre på å klassifisere treningsdataen, men *bare* treningsdataen, og potensielt dårligere på å klassifisere usett valideringdata. Kurven for Training Accuracy går mot 100% etter bare 10 epochs, som igjen tyder på at modellen er overfit.

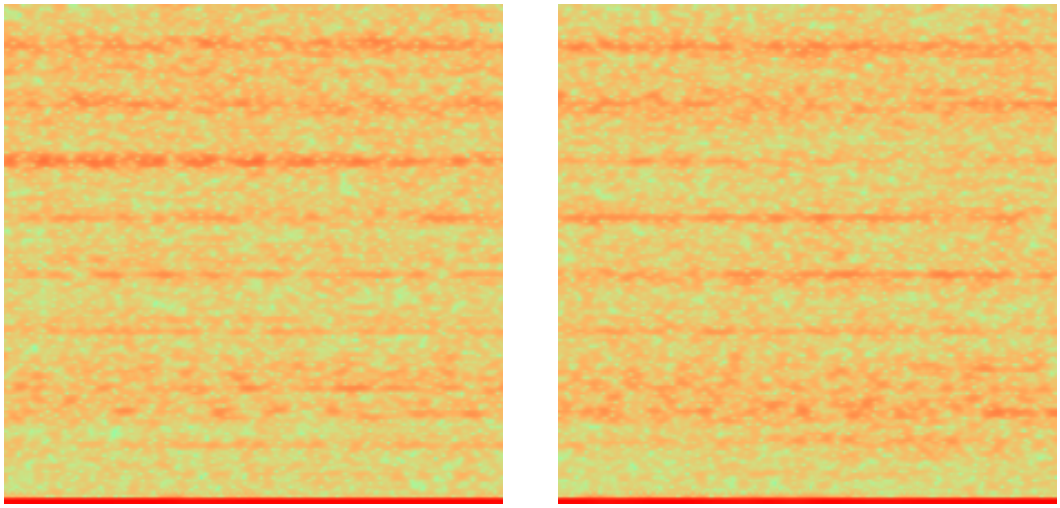
Det er noen små variasjoner mellom forsøkenes nettverk, blant annet at Validation Loss i forsøk 1 etter rundt 20 epochs har en svak negativ trend, samtidig som Validation Accuracy har en svak økning mellom 10 og 25 epochs. Det at Val Accuracy øker gjennom treningen og Val Loss øker i slutten av treningen kan bety at modellen blir bedre på å generalisere treningsdataen og derav presterer bedre. Likevel er trenings- og valideringsdaten ikke identisk, men veldig like, noe som gjør at den er veldig sammenlignbar med treningsdataen som kan føre til at den overfit modellen presterer greit på valideringssettet likevel. Dette vil som oftest gjelde alle modeller med et trenings- og valideringssett fordi ett datasett blir delt i to. Årsaken til at modellen er overfit er i dette tilfellet mest sannsynlig datasettet som brukes. I kapitlene som følger diskuteres dataen benyttet til trening.

5.1.1 Dataen

Begge modeller er trent ved hjelp av DeepShip-datasettet, der modellen i forsøk 1 er trent på hele datasettet, og i forsøk 2 på rundt 57% av DeepShip-datasettet.

Selv om datasettet består av over femti tusen ulike bilder består disse av spektrogrammer av kun 256 unike skip. Hvert klipp er delt opp i 3 sekunder lange lydklipp som deretter har blitt konvertert til spektrogrammer. Man kan argumentere for at teknikken

for dataaugmentasjon i praksis har laget to tusen "like" bilder av hvert av de 256 skipene. Figur 5.1.1 viser et eksempel på to spektrogrammer med 60 sekunders mellomrom og er veldig like. Det er små forskjeller, som en tykkere linje nummer tre fra toppen, likevel er det et eksempel som kan illustrere hvorfor modellene evner å memorere dataen etter få epochs.



(a) 20171119b-29Cargo - clip 13 **(b)** 20171119b-29Cargo - clip 33

Figur 5.1.1: To ulike spektrogram fra kategorien *Cargo* med 60 sekunder tidsintervall imellom.

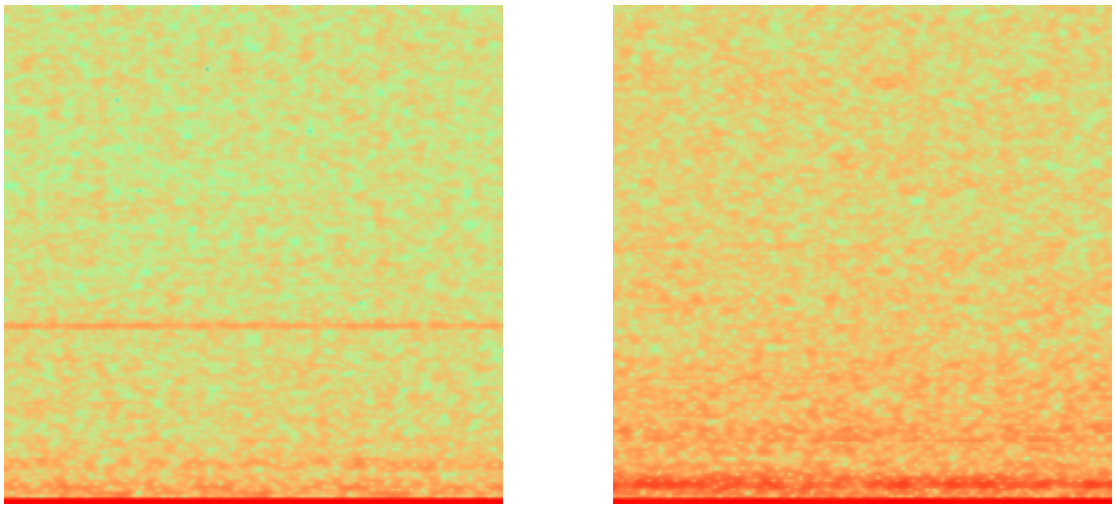
Konklusjonen er at modellen har fått dårlige forutsetninger til å generalisere dataen til de fire kategoriene fordi treningsdaten ikke har nok variasjoner i eksemplene. Modellene trenger flere eksempler for å ikke bli overfit, og eksemplene bør være mangfoldige. Mangfoldig data ville i dette tilfellet vært data som inneholder flere unike skipspasseringer og flere unike fartøyer.

Et tiltak som kan hjelpe mot overfitting er å regularisere modellen ved å legget til *weight decay* som vist i kodeeksempel 5.1. Regularisering gjør at modellen i mindre grad vil generalisere støy i dataen i stedet for de interessante frekvensmønstrene. *Weight decay* er en faktor som ganges inn i kostfunksjonen som reduserer vektene til parameterne. På denne måten får modellen i praksis færre parametere og dermed blir modellen mindre kompleks. Dette har ikke blitt testet i oppgaven men er en anbefaling for veien videre.

```
optimizer_ft = torch.optim.SGD(model_ft.parameters(), lr=0.001,
    momentum=0.9, weight_decay=1e-5)
```

Kodeeksempel 5.1: Forslag til endring som tilfører regularisering i modellen.

En annen kilde til at modellene er overfit er mengden støy i dataen. Med støy menes mengden informasjon i dataeksemplene som ikke er kjennetegnende eller relevant. Preprosesseringen av dataen tar ikke høyde for annet enn å lage mange klipp og å lage spektrogrammer ved bruk av STFT av den "råe" lyden. Resultatet av dette er at spektogrammene viser alle frekvensene tilstede i lydklippet inkludert store mengder støy som forstyrrer delene av signalet som er interessant. Nettverket trenes dermed opp til å gjenkjenne både frekvensmønstrene i kategoriene, men også støy som ligger i dataen. Hver piksel som inneholder annen informasjon enn det som er interessant for generaliseringen vil gjøre modellen litt mindre nøyaktig. Figur 5.1.2 vises to eksempler, et fra kategorien *Tanker* og en fra *Tug*, som inneholder svake antydninger til signaturfrekvenser som overdøves av støy. Spesielt figur 5.1.2(b) viser at lyden inneholder store mengder støy i form av at store deler av spektrogrammet har høye amplituder. Området opptakene til datasettet er tatt preges av stor skipstrafikk som potensielt gjør at frekvenser fra diverse støykilder blir fanget opp i dataen. I tillegg utgjør støy i sjøen en del av støybildet, som kan gjøre at modellen feilaktig plukker opp lokale karakteristikker. Det kan tenkes at dette gjør den trente modellen mindre anvendbar i et annet geografisk område fordi bakgrunnsstøyen vil være annerledes. Dersom modellen er trent til å generalisere støy og signaler sammen er det mindre sammenlignbart med data fra en annen lokasjon.



(a) 20161107-51Tanker - clip 43 **(b)** 20171120a-18Tug - clip 45

Figur 5.1.2: To spektrogram fra henholdsvis *Tanker* og *Tug*. Figurene viser støy i treningsdataen.

Datasettet bør være representativt, som i dette tilfellet kan bety å forsøke å eliminere

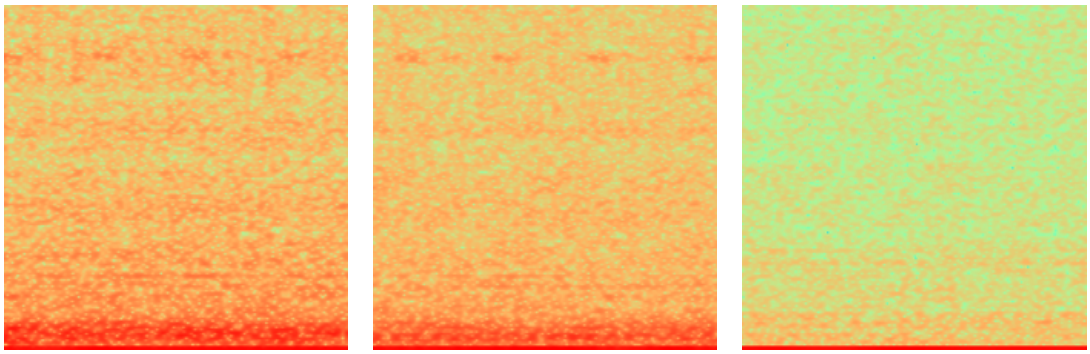
støy slik at kun ønsket karakteristikk kommer fram i bildene. En løsning på dette er å bruke et filter som demper uinteressante frekvenser og å fremheve frekvensene som kjennetegner fartøysignaturen. Optimalt bør dataen utelukkende inneholde frekvensene av interesse.

Et annet tiltak er å endre hvilken visuell fremstilling dataen skal ha. I følge Irfan et al., 2021 får ResNet en Val Accuracy på over åtte prosentpoeng høyere enn Mel-Spectrogram når datasettet er transformert med Constant Q Transform (CQT). De sistnevnte fremstillingene bruker en annen matematisk metode for å få fram frekvensene i lyden og kan implementeres med hensikt om å forbedre dataen.

Et annet alternativ kan være å vurdere andre fremstillinger av data enn et visuelt format som skal gjennom et konvulsjonsnett. Det kan også være et alternativ å benytte frekvensspekter for korte tidsintervaller i stedet for frekvens plottet over tid. Da ser modellen på frekvensene fra et kort øyeblikk, noe som kan øke responstiden til systemet sammenlignet med å bruke 3 sekunder lange klipp. Med dette menes at et system som klassifiserer lydsignaler fortløpende kan gjøre prediksjoner "tidligere" enn dersom man må vente på at det har gått 3 sekunder.

En annen støykilde i dataen er fravær av fartøyssignaturer i begynnelsen og slutten av skipspasseringer. Avhengig av avstanden fra hydrofonen til fartøyet endrer tidspunktet der fartøysstøy plukkes opp av hydrofonen og overstiger støynivået. Et fartøy som er nærmere hydrofonen vil oppfattes som høyere og mørkere i spektrogrammet enn et langt unna. Alle skipspasseringer er innenfor en 2 km radius, likevel vil det være variasjoner i lydstyrken som oppfattes. I mange skipspasseringer har de første og siste spektrogrammen lite til ingen fartøyssignatur, som fører til at kategoriene forurenses av data som ikke inneholder kategoriens karakteristikker. Figur 5.1.3 viser hvordan spektrogrammer fra samme passering kan variere i utseende. Figur 5.1.3(c) er det siste spektrogrammet i passeringer hvor det tyder på at fartøyet er langt utenfor rekkevidde. Konsekvensen av å ha med lignende bilder i datasettet er at kategoriene forurenses av irrelevant data. Et tiltak er å fjerne den andelen av spektrogrammene som ikke ser ut til å inneholde fartøysignatur.

Et annet tiltak som kan forbedre modellens prestasjon er å legge til avstandsinformasjon i modellens trening. I datasettets metadata er det informasjon om fartøyenes avstander til hydrofonen som ikke har blitt brukt i denne oppgaven. Det å legge til av-



(a) 20171120a-18Tug - clip 1 **(b)** 20171120a-18Tug - clip 21 **(c)** 20171120a-18Tug - clip 130

Figur 5.1.3: Tre spektrogram som viser endringene i løpet av samme passering.

stander som en egenskap *eng. feature* kan øke Val-Accuracy fordi da vil passeringer kjennetegnes av både frekvenser og lydnivå i dB.

Til slutt er det også verdt å nevne at det, i tillegg til regularisering, finnes flere variabler som kan justeres for å få et nettverk som presterer bedre på dataen. Likevel er nettverket overfit i så høy grad at det til tross for endringer i blant annet læringsrate vil utgjøre en liten forskjell i prestasjonen. Det er antakeligvis det å redusere nettverkets kompleksitet som kan påvirke modellen i stor grad. Et ResNet-18-nettverk har 18 lag, som betyr at den har veldig mange parametere å justere. Å redusere antall lag i nettverket kan redusere kompleksiteten og gjøre den bedre tilpasset dataen.

Det er mange tiltak som kan forbedre modellen og kan implementeres raskt, som endring i diverse variabler og å teste flere nettverk. Det å generere mer data tar lang tid, men vil forbedre enhver modell og anbefales sterkt. Det å lage passende filtre kan også forbedre modellen og gjøre den mer generaliserbar. Det er mangelen på data og effektiv preprossesering som er de største utfordringene for å lage en pålitelig ResNet-18-nettverk for auralklassifikasjon.

5.2 Testresultatene

Kapittelet har til nå sett på evaluering av nettverket uavhengig av testene i forsø 1 og 2. I dette delkapittelet blir testresultatene for forsøk 1 og 2 drøftet. I tillegg til nettverkets prestasjon er det andre faktorer som muligens påvirker testresultatene som er individuelle for de to forsøkene.

Resultatene fra forsøk 1 viser at nettverket har en test-Accuracy på 25.83%, som

såvidt er over den statistiske sannsynligheten for å velge riktig kategori av ren tilfeldighet, 25%. I praksis er en modell med så liten nøyaktighet ubrukelig i enhver sammenheng. Den lave verdien på test-Accuracy tilsier at nettverket er lite anvendbart til å klassifisere LoVe-dataen.

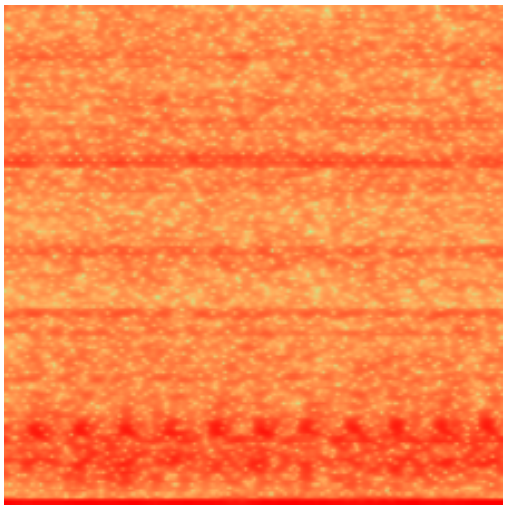
Data fra DeepShip og LoVe er preprossesert med helt lik fremgangsmåte. De største forskjellene mellom datasettene er at dataen er generert med to ulike sensorer og på ulik geografisk lokasjon.

Endring i geografisk lokasjon kan bety store endringer i undervannsmiljøet, som endring topologi og støy. Endring i undervannstopologi betyr i denne konteksten en stor endring i undervannsmiljøet i form av dybde, fjell og generell grovhet i havbunnen. Begrepet tar for seg store og små forskjeller, som for eksempel endringen fra å være midt i en norsk fjord til åpent grunt hav. Undervannakustikk er som kjent svært påvirket av reverberasjon som lager støy som et nevralt nettverk potensielt kan ha vanskelig for å skille ut.

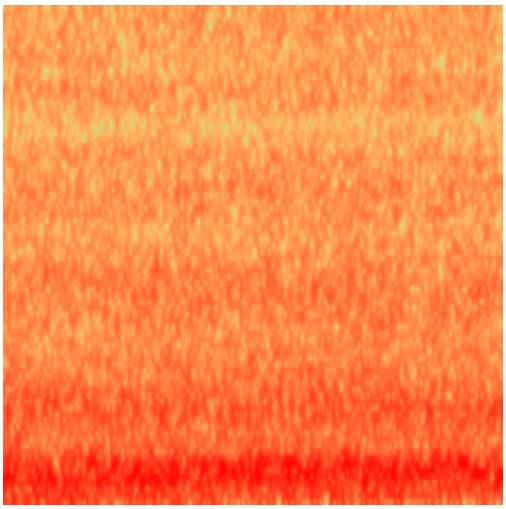
Endring i sensortype betyr at hydrofonene benyttet til å innhente data til to ulike datasett er forskjellige. I tillegg vil to hydrofoner i de fleste sammenligninger være plassert på ulik dybde, derav også ulike forhold mellom trykk, temperatur og saltnivå. Forskjellen på datainnhentingen til to hydrofoner kan være veldig forskjellig, der følsomheten, for eksempel på frekvensområder og forskjellige kalibrering, og samplingsfrekvensen er viktige faktorer som kan variere.

En av grunnene til at nettverket får en lav test-Accuracy er at bakgrunnsstøyen er så ulike mellom de to datasettene at modellen, som er overfit, gir bakgrunnsstøyen for stor vekt i treningen. Med andre ord klarer ikke nettverket å gjenkjenne treningsdataen i testdataen fordi den er trent til å gjenkjenne både fartøyssignatur og støy i kombinasjon. Figur 5.2.1 viser en sammenligning av to spektrogrammer markert som Passengership fra hvert av datasettene. For menneskeøyet er de to spektrogrammene veldig forskjellige, som også kan forklare at modellen presterer dårlig i sin prediksjon av kategori. Dette gjenspeiles av kategoriens F1-verdi på ca. 6%. Spektrogrammet til høyre er trukket frem fordi det viser det tydeligste "mønsteret" av testdataen i Passengership-kategorien. Spektrogrammet til venstre er vilkårlig valgt.

I tilegg har Passengership-kategorien en høyere Precision enn Recall, som tyder på at modellen predikerer *Passengership* sjelden nok at antall FN blir høy. Dette kan



(a) 20170111-52Passengership - clip 44 - DeepShip

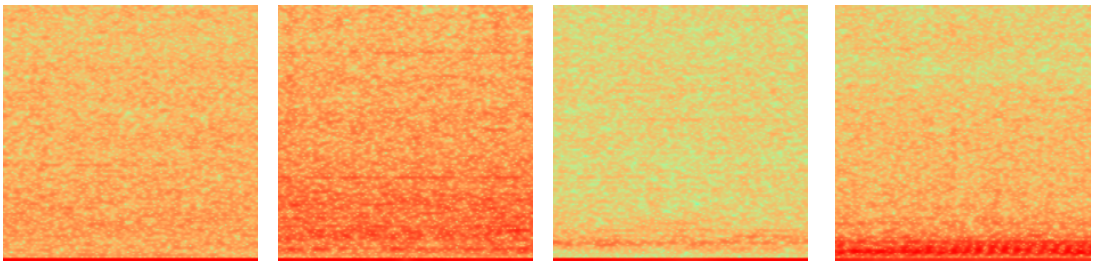


(b) 20190304-231500 - clip 7 - LoVe

Figur 5.2.1: To spektrogram, ett fra treningssett og ett fra testsett. Figurene sammenlignes for å vise ulikheten.

skyldes både endring i bakgrunnsstøy, men også at passeringene i kategorien har mye støy eller inneholder lite informasjon som karakteriserer kategorien. Det er i tillegg gjentakende for passeringer markert som Passengership at de første og siste spektrogrammene har veldig lite informasjon. En mulig årsak til dette er at lydnivået fra fartøy i denne kategorien er generelt lav. Dette er sammenlignbart med situasjoner i anti-ubåtkrigføring dersom ubåter har et lavere lydnivå enn hydrofonene kan oppfatte. Figur 5.2.2 viser fire eksempler på spektrogrammer fra kategorien Passengerships. Hvert spektrogram stammer fra hver sin passering. Eksemplene viser at flere spektrogrammer inneholder lite visuell informasjon som igjen er en forklaring på den lave verdien for F1-score. Spektrogrammene inneholder frekvenser i nedre del, som kan bety at verdimengden til spektrogrammene gjør at de lave frekvenser ikke blir fremhevet.

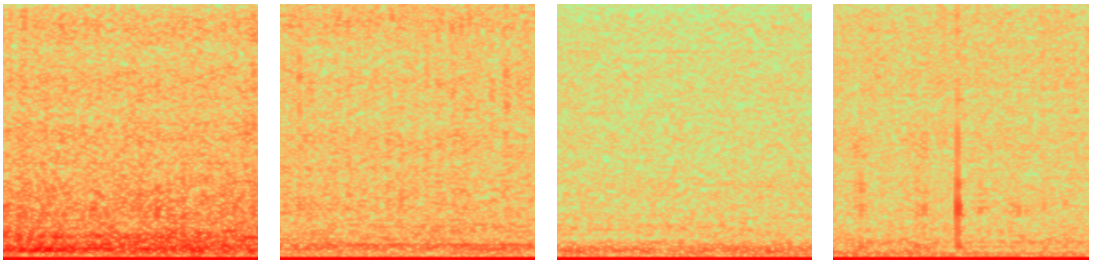
Det samme gjelder for kateogrien *Tanker*. Ved å se på treningsdataen preges mange spektrogram av støy og lite informasjon om fartøyet. Dette kan føre til at modellen klassifiserer alle spektrogrammer uten tydelige hovedfrekvenser og med mye støy som *Tanker*. Figur 5.2.3 viser fire eksempler annotert som *Tanker* i DeepShip-datasettet. Tanker oppnådde en Precision på over 63%, det høyeste av alle testeksemplene. Samtidig fikk Tanker relativt høy FP på 209. Selv om *Tanker* oppnådde en relativt høy verdi for TP er dette fordi modellen klassifiserer store deler av testsettet som *Tanker*. Sannsynligheten for å få høy Precision øker jo flere korrekte *Tanker*-prediksjoner som



(a) 20170322-60Passengership - clip 4 **(b)** 20170401-63Passengership - clip 1 **(c)** 20170408-65Passengership - clip 5 **(d)** 20170218-56Passengership - clip 68

Figur 5.2.2: Fire spektrogram fra ulike passeringer i kategorien *Passengership*s fra DeepShip om viser at kategorien inneholder "tomme" spektrogram.

blir gjort totalt fordi FP blir ignorert.



(a) 20180129-172Tanker - clip 71 **(b)** 20180203-174Tanker - clip 16 **(c)** 20180211-178Tanker - clip 76 **(d)** 20180221-179Tanker - clip 37

Figur 5.2.3: Fire eksempler fra ulike passeringer i kategorien *Tanker* som inneholder mange spektrogram uten lesbar informasjon.

Fartøyenes avstand til hydrofonen er ikke avklart i testeksemplene. LoVe-dataen er annotert med avstandsradius på rundt 20km. For å gjøre dataen sammenlignbar med DeepShip er passeringene trimmet til å inneholde lyd når fartøyene er 2km eller nærmere. Feilmarginen på denne avstanden er stor, noe som gjør at avstandsdataen i de fleste tilfeller ikke er nøyaktige, som fører til at lydnivåene varierer fra DeepShip-dataen i stor grad. Ønskelig bør avstanden mellom fartøy og hydrofon være lik i dataen som sammenlignes. I noen tilfeller kan det bety at man i praksis sammenligner to spektrogrammer der det ene fartøyet er lenger unna enn 2km og potensielt uoppfattet av hydrofonen. Avstand og amplitude (lydintensitet) er direkte knyttet gjennom formel 2.2. I mange tilfeller kan spektrogrammer feilklassifiseres fordi modellen har generalisert amplituden og støy i bildet. Både testdata og treningsdata bør være annotert med avstandsdata på hvert spektrogram for å dempe denne effekten og å knytte frekvens og

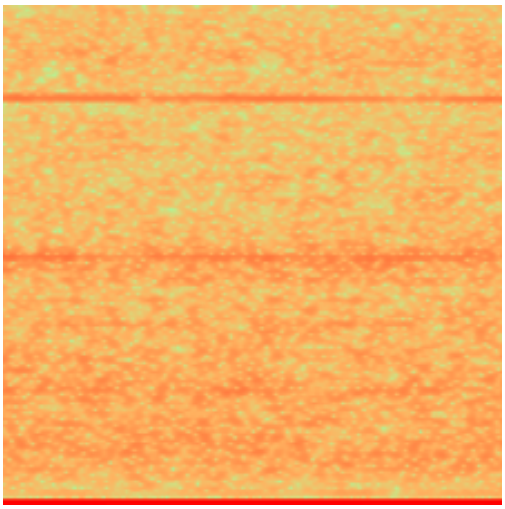
dB sammen i klassifikasjon av fartøy.

Selv om datasettet er relativt stort og hver kategori har rundt 10 000 dataeksempler tyder resultatene på at dataen ikke er representativ nok til at modellen kan kjenne igjen mønstre spesifikke til kategoriene. Dette kan forklares med at det kan være store variasjoner i fartøyssignatur innenfor hver kategori. I DeepShip-datasettet har hver kategori i snitt 64 unike fartøy representert, som sett opp mot resultatene tilsier at modellen ikke har lært seg hvordan fartøyssignaturene i hver kategori utarter seg uavhengig av geografisk lokasjon. Det samme gjelder testdataen, som har kun en passering i hver kategori foruten *Cargo* med sine to passeringer. En passering, derav ett fartøy, er ikke representativt for de mange ulike fartøyene i hver kategori nettverket kan møte på. Dermed er det sannsynlig at det innenfor minst en kategori benyttes et dårlig testeksempel.

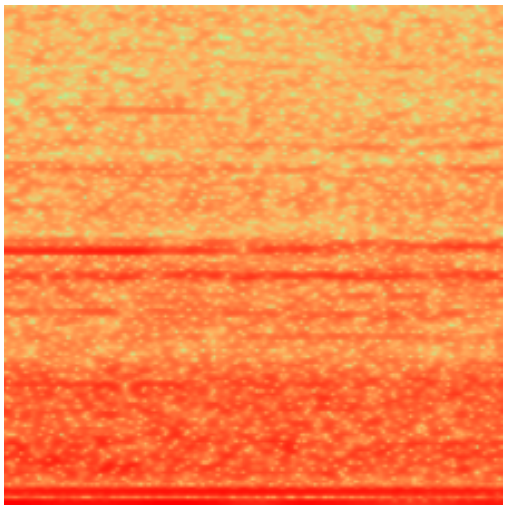
Kategorien *Tug,* som ble predikert i 8 av 750 prediksjoner, er et eksempel på at modellen ikke har forutsetning for å kjenne igjen *Tug* i testdataen. Ingen av prediksjonene av *Tug* er TP, muligens fordi mange av spektrogrammene i treningsdataen inneholder lite eller ingen karakteristisk informasjon. Ingen instanser av TP er det som fører til at Precision, Recall og F1-score til *Tug*-kategorien er lik null. I 189 av tilfellene av *Tug* predikerer modellen *Passengership* eller *Cargo*. Figur 5.2.4 viser sammenligningen mellom et *Tug*-eksempel fra treningssettet og testsettet. Begge spektrogrammene inneholder mye støy, og hovedfrekvensene er veldig ulike til tross for noen nærliggende linjer i midten og i øvdre del av spektrogrammet. Amplitudene til hovedfrekvensene er også veldig ulike. I sum kan dette forklare hvorfor nettverket unnlater å klassifisere høyre bilde som *Tug*. Eksemplet illustrer at nettverket i praksis er upålitelig ved klassifisering av data som divergerer fra treningssettet.

Det å øke antall passeringer i begge datasett kan forbedre modellens evne til å gjenkjenne mønstre basert på kategori i stedet for å memorere enkeltpasseringene i hver kategori.

Datasettene antas å være generert av to ulike sensorer. Informasjon om LoVe-sensoren er ukjent, men sannsynligheten for at hydrofonene er samme modell er lav grunnet at det er mange hydrofoner på kjøpsmarkedet. De viktigste forskjellene mellom sensorer kan være samplingsfrekvens og følsomhet. Førstnevnte kam gjøre at oppløsningen er ulik, men siden lyden konverteres til spektrogrammer er det visuelle forskjeller, spesielt i bakgrunnsstøy, som kan påvirke modellens prestasjon. Det finnes



(a) 20171223b-63Tug - clip 16 - DeepShip



(b) 20180508-041809 - clip 11 - LoVe

Figur 5.2.4: Sammenligning av spektrogram fra trenings- og testsett i kategorien *Tug*.

teknikker for å korellere samplingen slik at denne forskjellen elimineres.

Forsøk 1 er designet for å teste hvor godt nettverket trent på DeepShip-datasettet er til å klassifisere data fra LoVe-datasettet. Forsøket er ment å simulere et system som forflytter seg mellom to geografiske områder for å teste hvorvidt det er pålitelig idet undervannsmiljøet endrer seg. Resultatene viser at modellen utviklet for dette forsøket presterer nokså dårlig på å klassifisere data som har et annet opphav enn treningsdataen. Oppsummert er årsakene til dette en kombinasjon av de ovennevnte faktorene, inkludert at modellen er overfit. Det er antakeligvis på grunn av lite representativ data, forskjellig bakgrunnsstøy, unøyaktighet i avstander og ulike sensor er at modellen er dårlig på å klassifisere LoVe-datasettet. Forsøket viser at det med dagens datasett er utfordrende å utvikle en modell som evner å tilpasse seg endringer i undervannsmiljøet.

Nettverket i forsøk 2 er trent på passeringer fra vinterstid og får en Val Accuracy på ca. 68%. Nettverket er testet på kategoriene *Passengership* og *Tanker*. *Tug* og *Cargo* er ikke representert i testdataen. Som beskrevet er hensikten med forsøket å teste modellens evne til å klassifisere data fra en annen årstid. ResNet18-nettverket trenes på passeringer fra vintermåneder i DeepShip-datsettet og testes på passeringer fra sommermåneder. Resultatet gir er en test-Accuracy på 29.80%, som er såvidt høyere enn i forsøk 1. Det er enda en bekreftelse på at nettverket er overfit fordi den ikke klarer å generalisere treningsdataen. Det er andre faktorer enn selve modellen som påvirker resultatene, der noen har blitt diskutert i drøftingen for resultatene i forsøk 1. Spesielt

mangelen på representativ data er like relevant i forsøk 2.

Det er verdt å nevne at testdataen i forsøk 2 inneholder flere passeringer innenfor hver kategori. Dette styrker forsøkets resultater fordi testdaten er mer mangfoldig.

Siden testdataen stammer fra samme datasett som treningsdaten bør i teorien test-Accuracy ligne på Val Accuracy. Valideringsdataen er på samme måte som testdataen ny og usett data for modellen. Alle fullstendige passeringer er fordelt til enten trening, validering eller test-settene. En mulig forklaring er at bakgrunnsstøyen endrer seg etter temperatur og klima, som følgelig påvirker evnen til å klassifisere dataen. En annen forklaring er at Val Accuracy ikke tar høyde for variasjoner i kategorienes prestasjon, som kan være høy for noen kategorier og lav i andre.

Det er stor variasjon i mengden datapunkter i hver kategori som kan føre til at modellen trenes bedre på noen kategorier enn andre. *Tug* har 10 380 spektrogrammer i treningssettet, imens *Tanker* har kun 2543. Forskjellen gjør at dataen betegnes som ubalansert. Dette kan påvirke verdiene for Precision og Recall ved at noen av kategoriene er overrepresentert i testresultatene relativt til fordelingen av data i kategorier. Av alle tilfeller av *Passengership* ble *Cargo* og *Tug* predikert i over 80% av dem. Det er også *Cargo* og *Tug* som er overrepresentert i trenings- og valdideringssettene. En forklaring på dette kan være at systemet har større sjanse for å finne likheter mellom testdataen og kategoriene i treningsdataen som er mer mangfoldige. For å unngå at resultatene påvirkes for mye av ubalansen i data benyttes F1-score som balanserer verdiene for Precision og Recall basert på fordelingen av data i hver kategori.

Resultatene for *Tanker* skiller seg ut fra det ovennevnte med en test-Accuracy på over 50%. I motsetning til tidligere har *Tanker* et relativt lavt antall FP. Samtidig er dette et ufullstendig antall ettersom to kategorier ikke er representert i testdataen. Antakeligvis vil tester på de siste kategorien resultere i flere FP for både Passengership og Tanker.

Resultatene i forsøk 2 viser igjen at modellen egner seg lite til å klassifisere mellom fire fartøyskategorier i data som skiller seg ut fra treningsdaten. Årsakene kan være flere, men treningsdataen og preprosesseringen skiller seg ut som utløsende årsak til at modellen presterer dårlig på data fra en annen årstid. I tillegg kan prestasjonen forklares av at kategoriene Tanker og Passengership er underrepresentert.

5.3 Oppsummering

Testresultatene er preget av at nettverkene er overfit. Training Acc til begge nettverkene er veldig høy, nærmere 100%, som er en sterk indikator på at modellen memorerer testdataen i stedet for å generalisere den. Gjennom diskusjonen har det kommet frem at nettverkene trenger mer mangfoldig data for å gjøre dem mindre overfit. I tillegg bør preprosesseringen forbedres gjennom filtre som fremhever hovedsignalene og gjør dataen mer representativ. I forsøkene presterer nettverkene med en test-Accuracy på rundt 25-28%, som er veldig lavt. Den høye graden av overfitting og for få testeksempler gjør at nettverkene er generelt dårlig egnet til å klassifisere data fra en annen geografisk lokasjon eller årstid. For å forbedre nettverkene bør det produseres mer data av annoterte skipspasseringer med flere unike passeringer og flere unike fartøyer representert i dataen. Testresultatene kan også styrkes ved å inkludere flere skipspasseringer i testdataen. Oppsummert er det spesielt tilføying av relevant og mangfoldig data til datasettet og forbedring av preprosesseringen som vil kunne forbedre resultatene.

Man kan argumentere for at nettverkene i forsøk 1 og 2 er lite egnet til å anvendes data fra andre geografiske lokasjoner eller årtsider. Likevel er det flere tiltak som kan gjøre nettverkene i bedre stand til å generalisere dataen til å omfatte flere situasjoner. Konklusjonen er at det trengs større og mer mangfoldige datasett, i tillegg til bedre preprosessering for å lage et system som er pålitelig i variable miljøer. Sitatet *«the mapping function learned will only be as good as the data you provide it from which to learn" (Jason Brownlee, 2019),* er en godt erfart sannhet i denne oppgaven.

Konklusjon

Målsetningen til oppgaven har vært å anvende et kunstig nevral nettverkt til å klassifisere fartøy i hydrofonopptakene fra DeepShip-datasettet. I tillegg har intensjonen vært å teste hvorvidt nettverkene er anvendbare på andre datasett enn de er trent på. Oppgaven har bestått av å trene og teste to ResNet-18-nettverk i to individuelle forsøk. Hensikten har vært å teste hvor godt nettverkene presterer på data fra varierende geografisk lokasjon og årtsid.

Forsøk 1 har bestått av å trene et ResNet-18-nettverk på hele DeepShip-datasettet for så å teste den på passeringer fra LoVe-datasettet. Nettverket fikk en Val Accuracy på 68% med store antydninger til å være overfit. Testresultatene på LoVe-dataen gir en test-Accuracy på rundt 25%. Resultatene drøftes i mer detalj i kapittel 5.

Forsøk 2 omfatter trening av et ResNet-18-nettverk på DeepShip-data som har opprinnelse fra vintermånedene. Deretter testes nettverket på data, også fra DeepShip, som har opprinnelse fra sommenmånedene. Val Accuracy er på 68% og modellen vurderes til å være overfit. Testresultatene gir en test-Accuracy på 28%. De fullstendige resultatene er samlet i vedlegg C.

Resultatene i begge forsøk har gitt test-Accuracy mellom 25-28% som det kan argumenteres for at er *lavt.* Mulige årsaker til de lave resultatene diskuteres mer i dybden i kapittel 5. Diksusjonen har kommet frem til at de viktigste årsakene til nettverkenes prestasjon er at modellene er overfit, som betyr at den er overtilpasset treningsdataen og memorerer datapunktene. Det skjer når nettverket er for komplekst for dataen eller hvis treningssettet er altfor lite. Konklusjonen er at til tross for å bruke datasettet DeepShip, er også dette for lite til å generalisere fire fartøysklasser. I tillegg til å trenge mer data, er det like viktig at dataen er mangfoldig og representativ dersom et nettverk er tiltenkt å anvendes i varierende geografisk lokasjon og årstider. Et trent nevralt nettverk presterer

veldig høyt på data som ligner på treningsdataen, noe som gjør det utfordrende å lage allsidige nettverk, da det ikke finnes nok data som tar hensyn til den raske endringen i undervannsmiljøer. Oppsummert kan det argumenteres for at innhenting og annotering av relevant og mangfoldig data er et viktig tiltak for å kunne lage allsidige og pålitelige ML-systemer til militær bruk.

Bedre tilpasset preprosessering er et tiltak som kan forbedre nettverkenes prestasjon. I denne oppgaven har lydfilene blitt konvertert til spektrogrammer ved hjelp av STFT, og er en av mange måter å fremstille lyd grafisk. Det kan med fordel undersøkes om det går an å få frem karakteristikker i dataen ved bruk av en annen transformasjon. Et annet alternativ er å vurdere om bildegjenkjenning og konvulsjonsnett er den meste metoden for å klassifisere fartøyssignatur. Foruten det å fremstille lyddata som et bilde kan filtrering og bedre tilpasset bearbeiding få frem karakteristikker i dataen. Det kan gi nettverkene bedre forutsetning til å generalisere treningsdataen. Det å undersøke og teste ulike måter å forbedre preprosesseringen kan bidra til å utvikle et ML-system som presterer bedre.

De ovennevnte tiltakene er gjenkjennbare som vanlige utfordringer i utviklingen av pålitelige ML-systemer. Uten tilstrekkelig og representativ data kan ikke modellen generalisere problemstillingen, og uten et godt testet og tilpasset nettverk vil nettverkets prestasjon være høy nok. Spesielt i en militær kontekst, der risikoen er høy og marginen for feil er liten, er pålitelige systemer spesielt viktig. Om bord militære fartøyer stilles det høyre krav til systemenes robusthet og presisjon. Et ML-system må være nærmere feilfritt for å kunne overta oppgaver fra mennesker, noe oppgaven antyder at er langt fra tilfellet i anvendelsen av ML til auralklassifisering.

Det er ikke bare om bord ubåter og andre fartøyer som kan ha bruk for et slikt system. Dersom utviklingen av en algoritme for å gjenkjenne fartøytypers undervanssignatur lykkes i framtiden kan det blir et verdifullt verktøy i flere sammenhenger. Et selvstendig gjenkjenningsverktøy plassert nedover Norgeskysten kan bidra til overvåkingen av våre havområder. Et annet bruksområde er å plassere et slikt system på autonome farkoster med hensikt om å øke rekkevidden for overvåking. I tillegg kan et system som driver automatisk datainnhenting supplere til et slikt ML-verktøy og øke nøyaktigheten. Maskinlæring er i tillegg tilpasningsdyktig og kan trenes til å gjenkjenne ny fremdriftsteknologi.

Oppsummert har oppgaven anvendt et kunstig nevral nettverk til å klassifisere fartøyer i undervannsopptak og testet disse på ny usett data fra en annen geografisk lokasjon og årstid. Konklusjonen er at systemet presterer dårlig på testdata som varierer mye fra treningsdataen og er lite anvendbar i variable undervannsmiljøer. Likevel ligger det mye potensiale i det å utvikle konspetet videre fordi det i restn av verden satses mye på kunstig intelligens. Det å utvikle systemer som kan gi fordeler mot potensielle fiender er verdt å forske på og å investere i. KI sin rolle i militær teknologi er forventet å øke, noe som gjør at konseptet for et slikt system er relevant i dag.

6.1 Anbefaling

Det å anskaffe mer hydroakustisk data av fartøyspasseringer er tidkrevende. For å få dette til anbefales det å samarbeide på tvers av instanser. Oppgaven om å lage store nok datasettanbefales og er verdt satsningen. Det er datasettet som er avgjørende for å få et fungerende system fordi det å tilpasse preprosessering og utviklingen av et nettverket er oppgaver som det allerede finnes kompetanse for å gjøre. Nedenfor listes noen problemstillinger som kan være interessante å utforske:

– Hvordan kan et datainnhentingsverktøy for hydroakustisk data lages?

– Hvordan kan avstandsdata implementeres i klassifisering av fartøyssignatur?

Referanseliste

Forsvarskommisjonen. (2023). *Forsvar for fred og frihet* (2023:14). Departementenes sikkerhetsorganisasjoner. Oslo. https://www.regjeringen.no/contentassets/8b8a7fc642f44ef5b27a1465301492ff/no/pdfs/nou202320230014000dddpdfs.pdf

Forsvaret. (2023). *Forsvarssjefens fagmilitære råd.* https://www.forsvaret.no/aktuelt-og-presse/publikasjoner/fagmilitaert-rad/bilder-og-video/Forsvaret-FMR-2023.pdf/_/attachment/inline/c9147b67-7913-48ef-ac78-e61a2805f9a0:fd23bf41d3431040024613dfb377c033d84e2796/Forsvaret-FMR-2023.pdf

Ulrich Jochheim. (2021, September). *China's ambitions in artificial intelligence in artificial intelligence*. European Parliament. Brussel. https://www.europarl.europa.eu/RegData/etudes/ATAG/2021/696206/EPRS_ATA(2021)696206_EN.pdf

DataFlair. (2021). *Advantages and disadvantages of machine learning language* [DataFlair]. https://data-flair.training/blogs/advantages-and-disadvantages-of-machine-learning/

Irfan, M., Jiangbin, Z., Ali, S., Iqbal, M., Masood, Z., & Hamid, U. (2021, November 30). DeepShip: An underwater acoustic benchmark dataset and a separable convolution based autoencoder for classification. In *Elsevier* (p. 12, Vol. 183). https://www.sciencedirect.com/science/article/abs/pii/S0957417421007016?via%3Dihub

Gimse, H. (2017, April). *Classification of marine vessels using sonar data and a neural network* [Master of Science in Computer Science]. Norwegian University of Science and Technology. https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2453247/16334_FULLTEXT.pdf?sequence=1

Heimli og Vegusdal. (2018). *"sonaroperatøren" - auralklassifisering ved hjelp av et kunstig nevral nettverk* [bacheloroppgave]. FHS - Sjøkrigsskolen.

Waite, A. D. (2002). *Sonar for practising engineers* (3rd ed) [OCLC: ocm47271347]. Wiley.

Collimator. (2023, July 26). *What is a short-time fourier transform?* Retrieved November 29, 2023, from https://www.collimator.ai/reference-guides/what-is-a-short-time-fourier-transform

Gupta, P. (2017, November 16). *Regularization in machine learning* [Medium]. Retrieved December 1, 2023, from https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

Petter Brækken. (2005). *Lyd – en innføring*. Høgskolen i Sør-Trøndelag. http://brakken.no/hoyttalerteknologi/utdelt/Innforing_om_lyd_v3.pdf

Mark, T. (2021, March). *Examination of statistics and modulation of underwater acoustic ship signatures* (Scientific Report No. DRDC-RDDC-2021-R027). DRDC – Atlantic Research Centre. https://cradpdf.drdc-rddc.gc.ca/PDFS/unc358/p812955_A1b.pdf

T. Editors of Encyclopaedia. (2023, September 22). *Cavitation* [Britannica]. https://www.britannica.com/science/cavitation

Aural. (n.d.). In *Cambridge dictionary*. Cambridge. https://dictionary.cambridge.org/dictionary/english/aural

Tidemann, A., & Elster, A. C. (2023, July 26). maskinlæring. In *Store norske leksikon*. Retrieved October 12, 2023, from https://snl.no/maskinl%C3%A6ring

Brown, S. (2021, April 21). *Machine learning, explained* [MIT management]. Retrieved October 12, 2023, from https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

Tidemann, A. (2023, August 23). kunstig intelligens. In *Store norske leksikon*. Retrieved October 12, 2023, from https://snl.no/kunstig_intelligens

Géron, A. (2023). *Hands-on machine learning with scikit-learn, keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (Third edition). O'Reilly.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org

Buettgenbach, M. H. (2021, November 15). *Explain like i'm five: Artificial neurons* [Medium]. Retrieved November 29, 2023, from https://towardsdatascience.com/explain-like-im-five-artificial-neurons-b7c475b56189

Rastogi, V. (2023, September 8). *Fully connected layer* [Medium]. Retrieved November 29, 2023, from https://medium.com/@vaibhav1403/fully-connected-layer-f13275337c7c

Abueidda, D., Lu, Qiyue, & Koric, Seid. (2020). Deep learning collocation method for solid mechanics: Linear elasticity, hyperelasticity, and plasticity as examples.

Jason Brownlee. (2020, August 20). *A gentle introduction to the rectified linear unit (ReLU)*. A%20Gentle%20Introduction%20to%20the%20Rectified%20Linear%20Unit%20(ReLU)

Raschka, S. (2023, November 22). *Why is the ReLU function not differentiable at x=0?* [Sebastian raschka, PhD]. Retrieved November 29, 2023, from https://sebastianraschka.com/faq/docs/relu-derivative.html

Gaurav Singhal. (2020, May 5). *Transfer learning with ResNet in PyTorch | pluralsight*. Retrieved November 29, 2023, from https://www.pluralsight.com/guides/introduction-to-resnet

Saulo Barreto. (2023, June 16). *What is fine-tuning in neural networks?* [Baeldung]. https://www.baeldung.com/cs/fine-tuning-nn

Caulfield, B. (2009, December 17). *CPU vs GPU: What's the difference?* [NVIDIA blog]. Retrieved November 6, 2023, from https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

Heller, M. (2022, September 16). *What is CUDA? parallel programming for GPUs* [InfoWorld]. Retrieved November 25, 2023, from https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html

Ozechi, S. (2021, January 31). *4 techniques of evaluating the performance of deep learning models using validation.* [pyDataScience]. Retrieved November 25, 2023, from https://medium.com/pydatascience/4-techniques-of-evaluating-the-performance-of-deep-learning-models-using-validation-35ef4b12e8d8

SUSE. (2023). *What is a computer cluster? | answer from SUSE defines* [SUSE defines]. Retrieved November 6, 2023, from https://www.suse.com/suse-defines/definition/computer-cluster/

Riga Technical University. (n.d.). *HPC* [High performance computing center]. Retrieved November 29, 2023, from https://hpc.rtu.lv/hpc/?lang=en

SSH. (2023). *What is SSH (secure shell)? | SSH academy*. Retrieved November 29, 2023, from https://www.ssh.com/academy/ssh

Bright Computing. (2023, September 27). *User manual*.

eX3. (2023). *Resources* [eX3]. Retrieved November 20, 2023, from https://www.ex3.simula.no/resources

Python Software Foundation. (n.d.). *What is python? executive summary* [Python.org]. Retrieved November 21, 2023, from https://www.python.org/doc/essays/blurb/

NumPy Developers. (2023). *What is NumPy?* Retrieved November 21, 2023, from https://numpy.org/doc/stable/user/whatisnumpy.html

Jason Brownlee. (2019, May 23). *How much training data is required for machine learning?* [Machine learning mastery!]. https://machinelearningmastery.com/much-training-data-required-machine-learning/

Ocean Networks Canada. (2023). *Locations* [Ocean networks canada]. Retrieved November 26, 2023, from https://www.oceannetworks.ca/

Elle, P. J. (2017, November 3). *Havet utenfor Lofoten er mer variert enn i Nordsjøen*. Retrieved November 21, 2023, from https://www.forskning.no/partner-hav-og-fiske-nord-universitet/havet-utenfor-lofoten-er-mer-variert-enn-i-nordsjoen/314782

LoVeOcean. (2023). *About LoVe* [LoVeOcean]. Retrieved November 26, 2023, from https://loveocean.no/about-love

Sasank Chilamkurthy. (2023). *Transfer learning for computer vision tutorial* [PyTorch]. Retrieved November 20, 2023, from https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

Antonino Ingargiola et al. (2015). *1. what is the jupyter notebook?* Retrieved November 20, 2023, from https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

Stanford Vision Lab. (2020). *ImageNet* [ImageNet]. Retrieved November 20, 2023, from https://www.image-net.org/

Matthew Inkawhich. (2023). *Saving and loading models — PyTorch tutorials 2.1.1+cu121 documentation*. Retrieved December 1, 2023, from https://pytorch.org/tutorials/beginner/saving_loading_models.html

Vedlegg

A - Kildekode

All kode benyttet i denne oppgaven kan aksesseres fra GitHub Repository eller via lenken:
https://github.com/biosbios/Bachelor_Stavland

Kildekoden består av:

A.1 Copy_to_one_dir.py Kopierer alle .wav-filer i alle mapper til en felles mappe.

A.2 Make_3sec_clips.py Klipper alle .wav-filer til klipp på 3s.

A.3 Make_multiple_spectrograms.py Fra .wav-filer til spektrogrammer.

A.4 Run_job.sbatch Eksempel på .sbatch-fil.

A.5 Test_script.ipynb Scriptet for å teste nettverk.

A.6 Training_script.ipynb Script for å trene nettverk.

A.7 LaTeX_code.zip Script for å trene nettverk.

A.8 transfer_learning_tutorial.ipynb Original treningsscript av Sasank Chilamkurthy.

For tilgang til datasettene ta kontakt med fortfatter.

B - eX3

B.1

Vedlegget gir en kort fremgangsmåte for å få tilgang til og benytte seg av maskinklyngen eX3 OBS: kan endre seg, sjekk nettsiden eller få hjelp av eX3.

Be først om tilgang direkte til eX3. Send en e-post til ex3-contact@simula.no for å skaffe en «sponsor»/project manager som er nødvendig for å fylle ut spørreskjema. Fyll ut spørreskjemaet. Så fort man har en bruker kan man logge seg inn via ssh med kommandoen:

```
$ ssh -YAC2 username@dnat.simula.no -p 60441
```

Når man har en bruker i serveren er det neste steget å overføre filer fra lokal PC til eX3-brukerserveren ved å bruke kommando:

```
$ rsync -avrz -e "ssh -p 60441" <filename> username@dnat.simula.no:
<destination path>/
```

Siden en maskinklynge kan ha mange brukere er det viktig at ikke alle kan kjøre tunge oppgaver samtidig fordi det vil ta opp all kapasiteten til systemet. Slurm er en måte å allokere riktig mengde ressurser til riktig tid. Overordnet består metoden for å kjøre en Jupyter Notebook på en maskinklynge av følgende steg:

1. Klargjør data, sbatch-fil og treningsscript på lokal PC (eller maskinklyngen) og overfør som vist over.
2. Kjør sbatch-fil i terminal og få allokert ressurser
3. Overvåk job
4. Overfør resultater fra server til lokal PC

Sbatch er en filtype som inneholder nødvendig informasjon om gjeldende "Job". Den inneholder hvilke operasjoner brukeren ønsker at noden skal gjennomføre. For å kjøre en .ipynb fil uten å bruker browser filen konverteres til filtypen .nbconvert før den kjøres. Kommandoen er:

```
$ srun jupyter nbconvert --to notebook --execute filename.ipynb
```

Sbatch-filen må inneholde de nødvendige modulene som skal lastes inn. Det finnes mange nyttige ressurser om slurm og sbatch på internett. I tillegg til datasettene og sbatch-filen trengs et treningsscript. Et eksempel på en sbatch-fil er:

```
#!/bin/bash
#SBATCH --job-name="try2"
#SBATCH --partition=dgx2q
#SBATCH --time=0-04:00:00
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --output=%j-%x-stdout.txt
#SBATCH --error=%j-%x-stderr.txt

module load cuda11.8/toolkit/11.8.0
module load cmake/gcc/3.27.4
module load openblas/dynamic/0.3.24
module load gcc11/11.3.0
module load python39
module load cudnn8.6-cuda11.8/8.6.0.163
module load hdf5_18/1.8.21 python39
module load ml-pythondeps-py39-cuda11.8-gcc11/4.12.0
module load nccl2-cuda11.8-gcc11/2.16.5
module load pytorch-py39-cuda11.7-gcc9/1.13.0
module load anaconda3

srun jupyter nbconvert --to notebook --execute filename.ipynb
```

Når alle script og data er overført til eX3-server kan .sbatch-filen kjøres med følgende kommando:

```
$ sbatch filename.sbatch
```

Underveis i kjøringen kan oppgaven overvåkes via flere kommandoer, noen eksempler er:

```
$ squeue
$ sview
```

Siden resultatene lagres i .ipynb må de overføres fra eX3-server til lokal pc. Det samme gjelder for .pt-filen som inneholder den ferdigtrente modellens parametere. For å hente filer fra serveren til lokasjonen "." benyttes SCP gjennom kommandoen:

```
$ scp -P 60441 username@dnat.simula.no:<filepath> .
```

I tillegg til de mest relevante kommandoene er det flere nyttige kommandoer som kan være til hjelp, men som ikke har blitt brukt i oppgaven. Blant annet er det måter å åpne Jupyter Notebook i serveren gjennom lokal browser. I eX3-terminalen kjøres Jupyter Notebook gjennom for eksempel port 8890:

```
$ jupyter notebook --no-browser --port=8890 &
```

I lokal terminal åpnes en tunell via SSH og port 8890:

```
$ ssh -t -L 8001:localhost:8001 brukernavn@dnat.simula.no
-p 60441 ssh -L 8001:localhost:8890 g001
```

Her er g001 navnet til noden som brukes. Det er mulig å teste diverse på noden uten å kjøre en -sbatch-fil. Denne metoden er strengt kun for testing fordi det å kjøre tunge oppgaver kan forstyrre andre brukere. Kommandoen for å logge inn på noden er:

```
$ srun -p dgx2q -N 1 -n 8 --gres=gpu:1 --pty /bin/bash --login
```

Mer informasjon om bruk av eX3 ligger i Vedlegg B.2.

B.2

Lenke til pdf-fil laget av eX3 i 2022 med mer informasjon:

https://www.jottacloud.com/s/3455dfa45352b7c4a0a80bcd2806659023c

C - Resultater

Resultatene er fremstilt i tabell .0.1 og .0.2. Flere resultater som inkluderer prediksjonsverdier ligger lagret digitalt i tekstfiler for hvert forsøk som er listet nedenfor:

- C.1 GeoResultaterCargo2017.txt
- C.2 GeoResultaterCargo2019.txt
- C.3 GeoResultaterPassengership.txt
- C.4 GeoResultaterTanker.txt
- C.5 GeoResultaterTug.txt
- C.6 VærResultaterTankerJuly.txt
- C.7 VærResultaterPassengershipJuly.txt

Filene kan aksesseres gjennom lenken:
https://www.jottacloud.com/s/3459b6dd58e3af741a68ed9556bd304a7e3

Resultater forsøk 1				
Kategori	TP	FP	TN	FN
Cargo	56	106	375	174
Passengership	11	230	401	109
Tanker	128	209	369	72
Tug	0	10	541	200
Totalt:	315	555		

Tabell .0.1: Tabell av testresultater for forsøk 1, sammenligning av DeepShip og LoVe-data. Resultatene inkluderer antall TP, FP, TN og FN for hver kategori og totalt.

Resultater forsøk 2				
Kategori	TP	FP	TN	FN
Cargo	0	625	650	0
Passengership	110	5	805	815
Tanker	407	464	868	403
Tug	0	531	1204	0
Totalt:	517	1625		

Tabell .0.2: Lik tabell som .0.1 for forsøk 2, kun DeepShip data.

D - Bruk av OpenAI

Alle script utenom trenings- og testingsscriptet er hentet fra OpenAI sin ChatGPT. Deretter har koden blitt modifisert spesifikt til oppgaven. I noen tilfeller er koden mer modifisert enn i andre. I tilfellene der koden er modifisert vil kommentarene i scriptet oppgi dette. For å illustrere hvordan koden er innhentet er spørringer listet i dette vedlegget. Spørringene er delt inn etter hvilket script det gjelder og er listet nedenfor:

Copy_to_one_dir.py

- *"make python script that copies all the wav files in all the folders of a directory to a new directory called "Cargo all clips". Call all the clips the same as their folder name"*
- *"modify the code so that all the clips are copied into the same folder"*
- *"make a script in python that goes through all folders and copy all wav files in each folder to a collective new folder for all of the clips. name alle the clips as their parent directory"*

Make_3sec_clips.py

- *"python code to make a function that splits a wav file into 3 sec clips and save all the clips in a folder"*
- *"change the code to applying the split_wav_into_clips function to all wav-files in a folder and then naming the cut clips the original filename- clip nr."*
- *"make sure all the new clips are stored in a new folder called "3_sec_clips""*

Make_multiple_spectrograms.py

- *"python code that takes a .wav file and creates a spectrogram from it"*
- *"can you increase the resolution of the spectrogram"*
- *"i want to change the colour theme"*
- *"i want to make the spectrogram without an axis"*
- *"i only want it to be saved without plotting to screen"*
- *"modify code to include a counter"*
- *"change code to only show frequencies 0-10kHz"*
- *"modify code so i can change size to 224x224"*

Ressursene er hentet i flere sesjoner som inneholder flere spørringer og informasjon enn det som ble benyttet i oppgaven. Sesjonene kan aksesseres gjennom lenkene:

- https://chat.openai.com/share/a4ae3167-258a-40d1-9e66-6d506dce09d4
- https://chat.openai.com/share/0c38e045-fe0c-4bda-99df-8e83b708864b

E - Preprosessering

Vedlegget inneholder detaljer om preprosesseringen.

Flere steg ble implementert for å tilpasse scriptet *Make_multiple_spectrograms.py*. Disse endringene listes nedenfor:

1. Endring av n_{fft}

Variabelen n_{fft} ble endret fra 512 til 4096 for å øke oppløsningen og å fremheve områdene med høy amplitude. Resultatet av å endre n_{fft} er at hovedfrekvensne blir tydeligere av at kontrasten øker. Følgende linje i scriptet redigeres for å endre verdien på n_{fft}:

```
n_fft = 4096  # Number of FFT points
```

2. Fjerning av akser og rutenett

Det er viktig at spektrogrammene er laget slik at detaljer, linjer og akser som er urelatert til dataen fjernes. I de fleste tilfeller betyr dette å fjerne alt av rutenett, akser og titler. Dette gjøres enkelt med følgende kodelinje:

```
axis ('off)
```

3. Eksperimentering med fargekode

Fargekoden benyttet heter *"rainbow"* .Fargekoden ble endret for å finne en fargekombinasjon som lager størst kontrast mellom høye og lave amplituder, derav tydelige kjennetegn i dataen. Fargekoden kan endres i linjen der spektrogrammet plottes med følgende endring:

```
spectrogram, _, _, _ = plt.specgram(audio_data, NFFT=n_fft,
Fs=sample_rate, noverlap=n_overlap, window=window,
cmap={"rainbow"})
```

Reskalering av bilder til 224 $\times$224

For å reskalere spektrogrammene til riktige dimensjoner blir følgende kodelinjer benyttet i scriptet i vedlegg A.3:

```
from PIL import Image

spectrogram_size = (224, 224)

img = Image.open(input_path)
img = img.resize(spectrogram_size, Image.LANCZOS)
img.save(output_path)
```